Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

Individual Fairness in Bayesian Neural Networks

Alice Doherty

April 17, 2023

A Final Year Project submitted in partial fulfilment

of the requirements for the degree of

B.A. (Mod.) in Computer Science and Business

Declaration

Alice Doherty

April 17, 2023

Permission to Lend and/or Copy

Individual Fairness in Bayesian Neural Networks

Alice Doherty, B.A. (Mod.) in Computer Science and Business

University of Dublin, Trinity College, 2023

Supervisor: Andrea Patanè

Abstract Deep learning models are increasingly being used to automate decision-making processes that can have a profound effect on the lives of individuals. Despite their benefits, it has been found that such models can display biases towards individuals based on sensitive attributes, such as gender or race.

In this project, we study Individual Fairness (IF) in Bayesian Neural Networks (BNNs). IF captures the notion that similar individuals should be treated similarly. BNNs are of particular interest, as it has been shown that BNNs display greater adversarial robustness than deterministic neural networks and although not the same, adversarial robustness has parallels to IF. Specifically, we adopt the ϵ-δ-IF definition, which states that, for any pair of ϵ-similar individuals, as determined by a given similarity metric, the output of a BNN should be δ-close.

We formulate this definition as an optimisation problem and propose the first method, to the best of our knowledge, for estimating IF in BNNs. We achieve this by adapting techniques used to verify adversarial robustness to be fairness-aware. Specifically, we introduce *Fair-FGSM*, an extension of FGSM to work with an appropriate similarity metric, that can be used to determine the worst-case treatment shown to an individual by a model.

We empirically evaluate our proposed method by using it to measure IF in a variety of BNNs of differing model architectures and similarity metric parameters, trained on a benchmark fairness task. We also conduct the same experiments on deterministic neural networks to serve as a point of comparison. Interestingly, we find that BNNs tend towards showing greater levels of IF in comparison to their deterministic counterparts.

Acknowledgments

First and foremost, I want to extend a huge thank you to my supervisor, Dr Andrea Patanè. This project would not have been possible without the guidance and support you have provided me with this past year. Thank you for sharing your knowledge, with such patience and enthusiasm.

To those I am lucky to call my friends, both old and new, thank you for making these past four years so memorable. You have each helped shape me into the person I am today and I will forever be grateful for that. Here's to many more years of laughter, tears, and adventures.

Finally, thank you to my family for putting up with me for the past 22 years. I could not have achieved what I have up until this point without you.

Alice Doherty

University of Dublin, Trinity College
April 2023

Contents

List of Figures

Listings

Acronyms

BNN	Bayesian Neural Network.
COMPAS	Correctional Offender Management Profiling for Alternative Sanctions.
FGSM	Fast Gradient Sign Method.
GAN	Generative Adversarial Network.
IF	Individual Fairness.
MCMC	Markov Chain Monte Carlo.
MILP	Mixed-Integer Linear Programming.
ML	Machine Learning.
NN	Neural Network.
VI	Variational Inference.
VOGN	Variational Online Gauss-Newton.

Chapter 1

Introduction

1.1 Context and Motivation

Machine Learning (ML) has gained popularity in recent years for its use in the automation of decision-making processes. The main motivating factors for ML's increased use include its ability to process complicated data much faster than humans, to consider far greater amounts of data, and to make objective decisions (Pessach and Shmueli, 2022). Additionally, ML algorithms have the advantage over humans as they do not tire, require far less time to become qualified decision-makers, and can potentially spot relevant patterns not obvious to humans (Barocas et al., 2019, Mehrabi et al., 2022). Some common examples of its use in high-stake decision-making processes include medical diagnostics (De Fauw et al., 2018), hiring processes (Bogen and Rieke, 2018), and lending decisions (Byanjankar et al., 2015). These examples all represent decisions that have a profound effect on humans at both an individual level and a societal level (Barocas et al., 2019, Oneto and Chiappa, 2020, Mitchell et al., 2020). Hence, it is imperative that these models are designed to make *fair* and *accurate* decisions.

Although ML models cannot express subjective biases like humans, models have been observed to show biases that come from either bias in training data or algorithmic design choices (Simonite, 2015, Dastin, 2018, Lambrecht and Tucker, 2019). One highly referenced example of the consequences of such models is that of Correctional Offender Management Profiling for Alternative Sanctions (COMPAS). COMPAS is a ML tool used to predict the likelihood of recidivism in the United States. A ProPublica investigation found that black people were more than twice as likely, compared to white people, to be predicted as high-risk but not go on to re-offend, whereas white people were more likely to be labelled low-risk but were then found to re-offend (Angwin et al., 2016).

Fairness in ML represents the area of research that investigates how to ensure ML algorithms do not make unfair decisions towards certain groups or individuals based on sensitive attributes, such as gender, race, sexual orientation, and disability, whilst still maintaining the benefits that come with algorithmic decision-making (Oneto and Chiappa, 2020). Although methods exist to evaluate fairness in state-of-the-art deep learning systems (Albarghouthi et al., 2017, Bastani et al., 2019, Benussi et al., 2022), there is a lack of fast and reliable methods to estimate fairness in uncertainty-aware deep learning algorithms. Algorithms that can model uncertainty allow for a distribution over the predictions to be returned rather than a single prediction (Abdar et al., 2021). This is especially important in the context of making fair and trustworthy decisions as it allows practitioners to determine how reliable an algorithm's predictions are and whether they can be trusted. Bayesian techniques offer one such solution to quantifying uncertainty in ML systems but the literature currently lacks the development of suitable methods to evaluate fairness in Bayesian deep learning models. This project aims to address this gap by developing a method to estimate fairness in Bayesian Neural Networks (BNNs).

1.2 Research Question and Project Objectives

Many definitions of what constitutes a fair ML model have been proposed, and we direct the interested reader to the reviews of Chouldechova and Roth (2018), Verma and Rubin (2018), Gajane and Pechenizkiy (2018), Friedler et al. (2019), Barocas et al. (2019), Mitchell et al. (2020), Pessach and Shmueli (2022), Mehrabi et al. (2022). Broadly, these definitions can be categorised into individual notions of fairness and group notions of fairness. This project will focus on the definition of Individual Fairness (IF) proposed by Dwork et al. (2012) which states that *similar individuals*, within a specific context, should be treated similarly. This differs from group fairness which captures the notion that *different groups* of individuals should be treated equally. Group fairness definitions often rely on enforcing some form of statistical parity (e.g. a model should output an equal number of positive classifications for members of protected and unprotected groups). However, there are many examples of when enforcing group fairness can lead to blatantly unfair outcomes at an individual level (Dwork et al., 2012), thus this project will focus on IF.

Dwork et al.'s (2012) IF definition has been widely accepted in the literature and methods exist to measure and improve IF in deterministic Neural Networks (NNs) (John et al., 2020, Yurochkin et al., 2020, Yeom and Fredrikson, 2020, Benussi et al., 2022). However, no such work exists in the context of BNNs. BNNs are stochastic NNs trained using Bayesian inference (Jospin et al., 2022) and are of interest in this context as research has shown that BNNs are more robust to gradient-based and gradient-free adversarial attacks when compared to deterministic NNs (Bortolussi et al., 2022). Although not the same, global adversarial robustness, which verifies a model's robustness to changes to any input in the domain (Katz et al., 2017), has many parallels to IF in which we verify that changes in an individual's sensitive features do not affect a model's prediction.

The relationship between notions of adversarial robustness and IF raises the question of whether the increased robustness of BNNs translates to an increase in IF. This has yet to be explored and in this project, we aim to add to the literature by firstly, developing a method to measure how individually fair BNNs are by transferring techniques from work done on verifying adversarial robustness, and secondly, by using the developed method to investigate whether BNNs are fairer than their deterministic counterparts, all other things equal.

The objectives of this project are summarised below.

- To develop a method to estimate IF in BNNs using adversarial examples. This involves extending current adversarial attack methods to work with a similarity metric, as is commonly used in the fairness literature.
- To empirically evaluate the resulting method by using it to estimate IF in a range of BNNs trained on a benchmark fairness task, and comparing the results to their deterministic counterparts.

1.3 Approach

In this project, we first develop a method to estimate IF in BNNs by transferring techniques used to verify adversarial robustness into the context of John et al.'s (2020) ϵ-δ-IF definition, introduced formally in Section 3.4.2. This definition formalises Dwork et al.'s (2012) IF definition using local perturbation constants and is presented as an optimisation problem. This formulation has been used to give guarantees of IF in fully-connected deterministic NNs using techniques such as Mixed-Integer Linear Programming (MILP) (Benussi et al., 2022). MILP ensures soundness and completeness, however, this approach has a worst-case exponential running time and cannot be used with BNNs. Instead, we propose the use of adversarial examples which present a more efficient and flexible solution to this optimisation problem.

We approach this by extending the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014), a widely used adversarial attack method, to work with BNNs and an appropriate similarity metric to measure the similarity of two individuals. We refer to this extended FGSM as *Fair-FGSM*. For every input (i.e. individual) x in our test space, we use *Fair-FGSM* to generate a corresponding adversarial example x', such that x' represents a similar individual, as determined by the similarity metric d_{fair}. Intuitively, a model's output for x' represents the prediction that would be made if x's sensitive attributes were able to vary arbitrarily, i.e. if the sensitive attribute has little to no influence on the output. Thus, by computing the difference in the resulting predictions for x and x' we can determine whether a model treats similar individuals similarly, and therefore if it is individually fair or not.

After establishing a method to measure IF in BNNs, we train several BNNs on the Adult dataset (Dua and Graff, 2017), a benchmark fairness task, and compare their IF scores to their deterministic counterparts. Additionally, we use models with varying architectures and different similarity metric parameters to provide better insight into how these variables may be affecting IF in BNNs.

1.4 Contributions

This project, to the best of our knowledge, is the first work that develops a method to evaluate IF in BNNs and that investigates how the probabilistic nature of BNNs can provide greater IF guarantees in the presence of biased data.

Specifically, the main contributions of this project are listed below.

- We develop a method to estimate IF in BNNs using adversarial examples. This is done by extending FGSM to work with a specified similarity metric in BNNs. These adversarial examples are generated such that every individual x has a corresponding similar individual x', as determined by an appropriate similarity metric, d_{fair} . By evaluating how a model treats similar individuals, we can determine how individually fair a model is. Formally, we adopt the ϵ-δ-IF definition proposed by John et al. (2020) into the context of BNNs which states that any pair of ϵ-similar individuals, measured by a similarity metric, should be treated δ-similarly. We use a modified FGSM, *Fair-FGSM*, to solve the proposed optimisation problem.

- We empirically evaluate our proposed method by using it to estimate IF in various BNNs trained on the Adult dataset (Dua and Graff, 2017). Through a series of experiments, using a variety of model architectures and similarity metric parameters, we show that BNNs display a tendency to be more individually fair in comparison to their deterministic equivalents.

1.5 Report Outline

This report is structured as follows. In Chapter 2 we present a review of the related work found in the literature. This is followed by Chapter 3 in which the background theory and notation required to follow this project are introduced. The method formulated in this project to measure IF in BNNs is then provided in Chapter 4. Chapter 5 describes the implementation of our method and the experiments carried out using our method to evaluate IF in a variety of BNNs and deterministic NNs. The results of these experiments are presented and analysed in Chapter 6, followed by a discussion of the limitations of this work. Finally, Chapter 7 concludes the report by summarising the work completed throughout this project and suggests potential areas for future work.

Chapter 2

Literature Review

In this chapter, a review of the existing work related to this project is presented. We begin in Section 2.1 by discussing the various sources of unfairness in ML systems. An overview of common fairness definitions is next presented in Section 2.2. Importantly, in this section, we distinguish between notions of individual and group fairness, the former of which is the notion used in this project. In Section 2.3 we then discuss the similarities between adversarial robustness and IF, giving a brief overview of existing methods that utilise techniques from adversarial robustness to mitigate unfairness in ML systems. In Section 2.4, we conclude this chapter by discussing the existing literature on measuring IF, highlighting the work that this project will build upon as well as the limitations of existing methods in the context of BNNs.

2.1 Sources of Unfairness

Despite the advantages of using ML to automate decision-making processes, biases in training data and algorithmic design can lead to individuals or groups being treated unfairly by these systems. In this section, we summarise the most common sources of bias in ML. We note, however, that the sources presented below are not exhaustive, and the reader is directed towards Mehrabi et al. (2022) for a comprehensive review.

2.1.1 Data Bias

ML algorithms by nature are data-driven and designed to learn patterns from data. Thus, it is not surprising that if the data used at training time is biased, the model may also display bias. Furthermore, the bias in the underlying data may not always be obvious or explicit. Some common sources of bias in data are listed below with references to examples of real-world algorithms that have been observed to project these biases.

Historical Bias Historical prejudice and discrimination are commonly reflected in the data collected on protected groups. Additionally, the predictions made by biased models further generate biased data, perpetuating historical biases. One such example of historical data bias leading to an unfair model is that of Amazon's ML recruiting tool designed to help automate the reviewing of job applications. Due to the historical dominance of males in the tech industry, this was reflected in its training data, leading to the tool inadvertently learning to be biased and discriminating against female applicants (Dastin, 2018).

Measurement Bias Decisions regarding how to translate observable features into measurable representations can introduce bias. The COMPAS algorithm, described in Section 1.1, is an example of where the unintentional use of certain proxies to measure an individual's risk for recidivism led to a model that was biased against racial minorities (Angwin et al., 2016).

Representation or Sampling Bias Sample selection bias can lead to the collection of unrepresentative data. For example, Buolamwini and Gebru (2018) show that benchmark image recognition datasets, such as IJB-B and Adience, consist mainly of light-skinned men. Their analysis found that this led to darker-skinned females having the highest misclassification rate, due to being overwhelmingly underrepresented in the data.

2.1.2 Algorithmic Bias

Algorithmic bias can be understood as a bias introduced solely by an algorithm and not as a feature of the input data. Design decisions, such as the choice of optimisation function, regularisations, and statistical estimators can all introduce algorithmic bias (Mehrabi et al., 2022). For example, if a ML algorithm is designed to minimise the overall aggregated prediction error, the algorithm will perform better on members of the majority group over the minority group (Pessach and Shmueli, 2022).

2.2 Fairness Definitions

To accurately measure the fairness of a ML algorithm, one must first decide what constitutes a fair decision. Complexity arises from the absence of a universal and objective measure of what is fair, and this is often context specific. Additionally, defining fairness is a problem that spreads far beyond computer science and into the social sciences and philosophy. Broadly, however, there is an agreement that a fair decision is one that is not influenced by the sensitive attributes of an individual or group (Oneto and Chiappa, 2020, Mehrabi et al., 2022). As already mentioned, the definitions of fairness presented in the literature can broadly be divided into two main categories: group fairness and individual fairness. In this section, an overview of both group and individual notions of fairness is given, however, the interested reader is pointed to the work of Chouldechova and Roth (2018), Gajane and Pechenizkiy (2018), Verma and Rubin (2018), Barocas et al. (2019), Friedler et al. (2019), Mitchell et al. (2020), Pessach and Shmueli (2022), Mehrabi et al. (2022) for thorough reviews of the fairness definitions presented in the literature.

2.2.1 Incompatibility of Fairness Definitions

The definition of fairness one employs must account for the context of the decision-making process. Critically, it has been shown that many definitions of fairness are incompatible, and thus it is not possible to simply combine various fairness metrics to get an all-encompassing definition (Kleinberg et al., 2016, Friedler et al., 2016, Corbett-Davies et al., 2017). Importantly, in the context of this project, it is not possible to combine both group and individual definitions of fairness without some form of trade-off (Dwork et al., 2012).

2.2.2 Group Fairness

Definitions of group fairness encapsulate the notion that different groups of individuals should be treated equally (Mehrabi et al., 2022). Most group fairness definitions are based on enforcing statistical notions such as independence, separation, and sufficiency (Barocas et al., 2019). Common group fairness definitions include disparate impact (Feldman et al., 2015), statistical or demographic parity (Dwork et al., 2012), equalised odds (Hardt et al., 2016), and equal opportunity (Hardt et al., 2016). For example, if sex is defined as a protected group, statistical parity classifies a decision that results in both males and females having an equal probability of a positive prediction as fair.

Group fairness definitions have been more widely adopted in the fairness literature and several methods have been developed to evaluate group notions of fairness in deterministic NNs. These approaches tend to rely on the probabilistic verification of group fairness definitions (Albarghouthi et al., 2017, Bastani et al., 2019). Additionally, methods to estimate group fairness in BNNs can also be found in the literature (Foulds et al., 2020, Perrone et al., 2021, Zeng et al., 2022). These rely on Bayesian optimisation techniques to tune a model's hyperparameters so that some definition of group fairness is achieved. Although these methods can be applied to measure fairness in BNNs, as they only consider group fairness they cannot be used in this project.

Furthermore, there are multiple publicly available fairness assessment tools that focus on measuring group fairness (Tramer et al., 2015, Galhotra et al., 2017, Zehlike et al., 2017, Bellamy et al., 2018, Saleiro et al., 2018). These, however, do not support measuring IF and are not compatible with BNNs. Therefore, they cannot be used in the context of this project.

Shortcomings of Group Fairness Despite the majority of the fairness literature focusing on group fairness, there are drawbacks to using group notions of fairness. For example, statistical parity will hold if the number of individuals within a protected group selected is proportional to those selected in the population (e.g. successful job applicants should be made up of 50 per cent women and 50 per cent men). It does not consider individual merit and therefore can reduce the utility of the decision and can also lead to explicit discrimination against individuals (Dwork et al., 2012). Overall, as group fairness relies on statistical notions, it may seem to provide fairness guarantees at a high level across groups, but it cannot give guarantees when every affected individual is considered. For this reason, we focus on measuring IF in this project.

2.2.3 Individual Fairness

In contrast to group fairness, individual notions of fairness are concerned with providing fairness guarantees at an individual level. In this section, we introduce two such notions: fairness through unawareness and fairness through awareness, the latter of which is the definition of IF utilised in this project.

Fairness Through Unawareness The simplest notion of IF is fairness through unawareness, also known as blindness. Fairness through unawareness states that an algorithm is fair if no sensitive attributes are explicitly used in the decision-making process (Gajane and Pechenizkiy, 2018, Mehrabi et al., 2022). An extension of this would be to assume that by removing the use of all sensitive attributes, one could design a fair and unbiased model. However, due to the presence of proxy attributes, this is not always the case. Proxy attributes are attributes that implicitly encode or are correlated to the value of a sensitive attribute. For example, an individual's zip code could be highly correlated with their race. Fairness through unawareness is often employed in situations where there is a legal requirement to not explicitly use sensitive attributes in a decision-making process (Oneto and Chiappa, 2020). However, it has been found that fairness through unawareness can often lead to less accurate models without significant improvements in the fairness of a model's outcome (Pedreshi et al., 2008).

Fairness Through Awareness Dwork et al. (2012) propose a definition of IF that addresses the insufficiency of fairness through unawareness. Fairness through awareness states that similar individuals should be treated similarly. In other words, a model is unfair if a pair of inputs (i.e. two individuals) are close to each other, as measured by an appropriate similarity metric, but are not predicted similarly by the model. For example, a model is unfair if two individuals who have similar credit scores are classified differently when applying for a loan. Importantly, the value of an individual's sensitive attributes should not affect their similarity to another individual. Due to the shortcomings of group fairness and fairness through unawareness described previously, this fairness through awareness notion of IF is the definition of fairness that is employed in this project. In this report, fairness through awareness will simply be referred to as IF as is common in the literature (Pessach and Shmueli, 2022).

Given the complexity of real-world applications, it is not always possible to capture the ground truth in the similarity metric and our best estimate must be used. This can represent a drawback to this definition of fairness as its value is heavily dependent on the accuracy of this metric. However, work has been done to create suitable metrics based on the judgement of qualified human arbiters (Ilvento, 2019) and automatically from data (John et al., 2020, Yurochkin et al., 2020, Mukherjee et al., 2020). Additionally, one could view the use of a task-specific similarity metric as an advantage of IF as it allows for expert and context-specific knowledge to be applied, rather than simply enforcing statistical parity, as is the case with group fairness, which may not reflect fairness in practice. Ideally, modular systems should be designed to allow for a similarity metric to be easily passed in. This would allow for the relevant policymakers to decide what is fair, rather than relying on engineers, with less knowledge of the sociological implications of their design decisions, deciding what constitutes a fair model. This project will rely on a weighted ℓ_p similarity metric, as described in Section 3.4.2, however, the potential for future work to extend the method presented in this project to support other metrics is discussed in Section 7.1.

2.2.4 Subgroup Fairness

As described in Section 2.2.1, it is not possible to combine individual and group fairness definitions and obtain the full benefits associated with each. Despite this, subgroup notions of fairness have been developed that attempt to enforce a combination of both group and individual fairness (Kearns et al., 2018, Hébert-Johnson et al., 2018, Speicher et al., 2018, Lohia et al., 2019, Mehrabi et al., 2022). However, it has been found that subgroup notions of fairness display the same shortcomings of group fairness, described in Section 2.2.2, as they still ultimately rely on considering average treatment across (sub)groups of individuals, albeit on a smaller scale (Chouldechova and Roth, 2018). Again, as we wish to provide guarantees for all affected individuals, we focus on measuring IF in this project.

2.2.5 Fairness-Accuracy Trade-Off

It is important to note that there is an inherent trade-off between a model's fairness and accuracy, such that as a model achieves greater fairness, its accuracy may suffer (Corbett-Davies et al., 2017). Consider a fairness-accuracy Pareto frontier on which every point corresponds to a model on the boundary of some optimal combination of fairness and accuracy. Any model not on this frontier can be considered bad, as it can be improved on either in accuracy or fairness without affecting the other measure (Kearns and Roth, 2019). More broadly, we can say the goal of a fair ML model is to maximise some notion of fairness without significantly impacting its accuracy. However, in the context of this project, it should be noted that in measuring the IF of BNNs we are not attesting to the accuracy of these predictions, and recognise that in some instances their accuracy may suffer when IF is enforced.

2.3 Adversarial Examples and Fairness

In this project, we propose a method to measure IF in BNNs that relies on the use adversarial examples. In this section, we aim to provide context for their use by introducing the similarities between adversarial robustness and IF, as well as through a brief description of how techniques from adversarial robustness have been used to improve the fairness of NNs in other work.

2.3.1 Relating Adversarial Robustness and Individual Fairness

Much work has been done to provide methods of verification for ML models (Liu et al., 2021). Adversarial robustness is one such method that requires small perturbations in input, known as adversarial examples, to not cause a significant effect on a model's output. Most work done in this area focuses on local robustness which verifies a model's robustness within a local neighbourhood of a given input (Katz et al., 2017). For example, given an input image, one checks that a small ℓ_∞-norm perturbation does not affect the prediction made by a classifier (Madry et al., 2017).

With IF, although similar, we wish to verify something closer to the notion of global adversarial robustness. Rather than considering a local neighbourhood, global robustness checks that perturbations of any input in the domain do not significantly affect a model's output. This in practice is much harder to verify and less work has been done on evaluating global robustness in the literature (Katz et al., 2017). Additionally, with robustness one tends to consider the average change in outputs, whereas, with IF we want to provide worst-case guarantees. This parallels one of the main distinctions between group fairness, which considers the average treatment of individuals across a group, and individual fairness, which considers the worst-case treatment of any one individual.

2.3.2 Adversarial-Based Fairness Enhancing Mechanisms

In Section 2.1 various sources of unfairness in ML were discussed, followed by Section 2.2 that explored the most common definitions of fairness in the literature. There is, however, another subarea of research yet to be discussed relating to the development of methods to increase fairness in ML models. These methods can be divided into three categories. Pre-processing methods modify the input data before it is fed into a model, in-processing methods modify the model itself, and post-processing methods, modify the output of a model (Pessach and Shmueli, 2022, Mehrabi et al., 2022). This represents a parallel line of work and is not directly related to this project, however, we wish to briefly highlight the use of adversarial learning as a fairness-enhancing mechanism. This is of interest as these techniques are based on methods originally designed to increase the adversarial robustness of ML models.

Generative Adversarial Networks (GANs) are ML algorithms used to generate realistic dataset samples (Goodfellow et al., 2020). These networks generally consist of a generator that continuously generates “fake” inputs and a discriminator that predicts whether the input is fake or real. A feedback loop is created, with the generator increasingly improving its ability to generate realistic samples, and the discriminator increasingly improving its ability to distinguish between real and fake samples (Goodfellow et al., 2020).

One such way GANs have been used to increase fairness is by using an adversary model to continuously check whether an input’s sensitive features can be inferred from a model’s prediction and updating the model accordingly (Zhang et al., 2018, Wadsworth et al., 2018, Celis and Keswani, 2019). Additionally, due to their ability to generate realistic input data, GANs have also been used to generate fair synthetic data that can be used to minimise the impact of underrepresentation and biased representation in training data (Xu et al., 2018, Zhang and Sang, 2020).

2.4 Measuring Individual Fairness

As previously mentioned, this project focuses on establishing a method to measure IF in BNNs. In this section, we discuss existing techniques that have been developed to measure IF, highlighting the limitations of these methods in the context of this project. We then introduce the relevant works relating to formulating and measuring IF that this project will build upon.

2.4.1 Existing Techniques for Evaluating Individual Fairness

As discussed in Section 2.2.2, there exist a number of methods to evaluate and improve group fairness in NNs. However, work on developing methods for IF is far more limited. Research exists in designing methods to train more individually fair models, including complex models such as NNs (McNamara et al., 2017, Yeom and Fredrikson, 2020, Yurochkin et al., 2020, Ruoss et al., 2020). These methods, however, have been developed for deterministic NNs and cannot directly be applied to BNNs. Additionally, although related, in this work we are concerned about evaluating the IF of a model, rather than improving it. Work does exist to translate Dwork et al.'s (2012) definition into IF verifiers (John et al., 2020) although this work is limited to linear and kernelized classifiers. Benussi et al. (2022) build on these verifiers to present the first work that provides global IF guarantees for NNs. However, their method relies on the use of MILP to compute these guarantees which is not suitable for use in BNNs. This project, however, will build on the work presented by John et al. (2020) and Benussi et al. (2022) as discussed in Section 2.4.2.

2.4.2 Individual Fairness as an Optimisation Problem

For simple linear models, Dwork et al.'s (2012) IF definition, described in Section 2.2.3, leads to a linear optimisation problem, and thus can be solved efficiently. However, for more complex models, such as NNs, the problem posed is non-linear and non-convex. John et al. (2020) propose a method to verify IF using logistic regression and kernel methods by solving this optimisation problem using MILP. Their work builds upon Dwork et al.'s (2012) IF definition to introduce the ϵ-δ IF definition which we adopt in this project. This states that two individuals are similar if $d_{\text{fair}}(x, x') \leq \epsilon$, where d_{fair} is an appropriate similarity metric, x and x' are the input features for a pair of individuals, and ϵ is an appropriate threshold. Predictions are considered similar if they are δ-close. This will be formalised in Section 3.4.2.

John et al. (2020) use MILP to solve the problem of whether a model displays individual bias according to the ϵ-δ IF definition, although this is limited to linear and kernalised classifiers. Additional work has been done to compute IF guarantees for fully-connected NNs using MILP (Benussi et al., 2022). MILP ensures soundness and completeness, however, this approach has a worst-case exponential running time and, crucially, cannot be used for BNNs. Adversarial attacks present a flexible and efficient solution to this optimisation problem, however, their use comes at the cost of not ensuring completeness. The exploration of the viability of using adversarial examples to measure IF is one of the core contributions of this project.

2.4.3 Similarity Metric

As described in Section 2.2.3, accurately measuring IF is heavily dependent on the choice of similarity metric used to capture the similarity between individuals. Ideally, the ground truth would be used but, in practice, this is often not possible. Ilvento (2019) propose various methods to generate an appropriate similarity metric based on the judgements of a human fairness arbiter. Additionally, work has been done to learn similarity metrics automatically from data (John et al., 2020, Ruoss et al., 2020, Mukherjee et al., 2020, Yurochkin et al., 2020).

This project will utilise a weighted ℓ_p metric to measure the similarity of individuals (John et al., 2020). This is defined formally as Equation (3.7) in Section 3.4.2. Broadly, however, it allows for weights of zero to be set for sensitive features (i.e. sensitive features should not affect a pair of individuals' similarity) and a context-appropriate threshold ϵ to be set for non-sensitive features.

The choice of this metric was motivated due to its use in similar works (John et al., 2020, Benussi et al., 2022), allowing for easier comparison of results. Additionally, compared to other metrics, it is less mathematically involved making it more suitable for this project in which we focus on the initial exploration of measuring IF in BNNs, rather than an exhaustive all-encompassing solution. However, we note that other techniques such as the Mahalanobis distance (Yurochkin et al., 2020) and feature embedding (Ruoss et al., 2020) serve as more flexible similarity metrics and can be used to more accurately account for proxies. We suggest, in Section 7.1, that the utilisation of these techniques should be explored in future work.

2.5 Summary

In this chapter, a review of the literature pertaining to this project was presented. First, a brief overview of how biases in data and algorithms can affect the fairness of ML systems was described. We then examined and contrasted group and individual notions of fairness. Critically, we highlighted that although group fairness methods have been more widely adopted, due to scenarios where their use can lead to blatantly unfair treatment at an individual level, IF is a more desirable notion. We then introduced the connection between adversarial robustness and IF, and the uses of techniques borrowed from adversarial robustness for improving fairness. Finally, existing work relating to evaluating IF was presented. Importantly, although there exist formulations that can be applied to verify IF in both linear models and more complex models, such as NNs, these cannot directly be used for BNNs. We conclude this chapter by describing the work that this project will build upon in order to develop a method to estimate IF in BNNs.

Chapter 3

Preliminaries

In this chapter, the theoretical concepts and notation used throughout this project are introduced. We begin in Section 3.1 with an introduction to NNs. In Section 3.2 we then discuss how Bayesian inference can be applied over a NN to obtain a BNN, and the key concepts associated with them. Adversarial examples, and the related concepts of adversarial attacks and adversarial robustness, are then formally introduced and discussed in Section 3.3. Finally, we formalise the definition of IF, briefly introduced previously in Section 2.4.2, that will be used in this project.

3.1 Neural Networks

ML is a subset of artificial intelligence that allows systems to learn by analysing patterns in data, rather than following hard-coded instructions (Goodfellow et al., 2016). Theoretical advancements in the field of ML, increased computational power, and increased data availability have given rise to the successful design and widespread use of NNs. NNs have the ability to learn relevant patterns from raw data offering more flexibility than traditional ML algorithms, which often require manual feature engineering.

3.1.1 Deterministic Neural Networks

At a high level, NNs are modelled on biological neural networks in the human brain. A NN is made up of layers, starting with an input layer, followed by one or more hidden layers, and ending with an output layer. Each layer consists of one or more nodes, also known as neurons, and the number of nodes in a layer is referred to as its width. Every node in a layer can be thought of as a function that takes in input data, has an associated weight and bias, and outputs a value. If the output of the node exceeds a given threshold, as determined by an activation function, the node is said to be activated and will send data to the next layer of the network (Goodfellow et al., 2016). The activation of a node in the input layer will depend on the input data provided. Nodes in the following layers will be activated according to the weighted values of previously activated neurons until the output layer is reached. As a model is trained, the performance of the model is calculated using a loss function, also referred to as a cost function. The goal at training time is to minimise this loss function by adjusting the weights and biases of the nodes in the model until a local minimum is reached. This is done through processes known as backpropagation and gradient descent (Schmidhuber, 2015).

Formally, we define a NN $f^w : X \to Y$, trained on a classification task and given an input set $X \subseteq \mathbb{R}^n$, an output set $Y \subseteq \mathbb{R}^C$, and a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$. Where $w \in \mathbb{R}^{n_w}$ is a combined vector of the model's weights and biases, n is the number of inputs, C is the number of output classes and N is the number of training points. For classification problems, given an input $x \in X$, $f^w(x)$ represents the class probability.

When we refer to deterministic NNs, we are referring to these standard NNs, which once trained will always produce the same output for a given input. This is in contrast to BNNs which are stochastic and are covered in detail in Section 3.2. An example of a deterministic NN is shown in Figure 3.1.

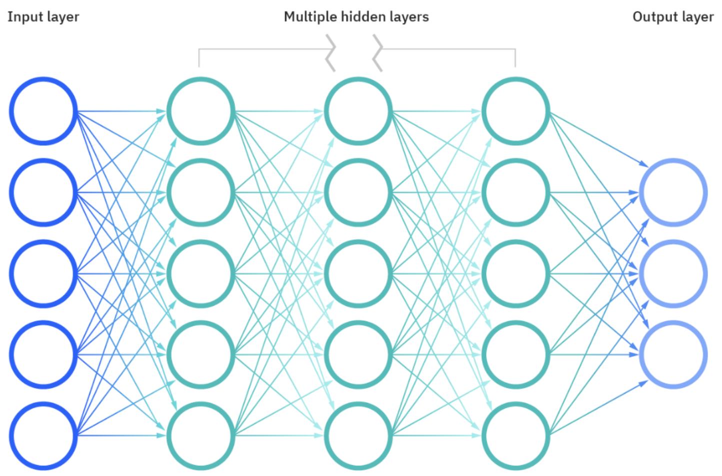

Figure 3.1: Diagram of a deterministic NN. This particular diagram shows a deterministic NN consisting of an input layer with 5 neurons, 3 hidden layers each with 5 neurons, and an output layer with 3 neurons. Image courtesy of IBM (n.d.).

3.1.2 Softmax Activation Function

The softmax function is used to convert a vector of n real numbers into a probability distribution, where the resulting probabilities must add up to 1. It is most commonly used as the activation function for the output layer of a NN dealing with multiclass classification problems (Goodfellow et al., 2016).

Formally, the softmax function $\sigma \colon \mathbb{R}^C \to (0,1)^C$ is defined as:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}, \tag{3.1}$$

for $i = 1, ..., N$ (Goodfellow et al., 2016).

Those with a greater familiarity with NNs may note that as this project solely deals with binary classification problems, it would be sufficient to use the sigmoid activation function. However, due to the nature of the code used to run our experiments, we have made the design decision to use the softmax function. We note that the softmax function is simply a generalisation of the sigmoid function to a multiclassification paradigm, and in our case, there is no difference in the results obtained by using either activation function (Goodfellow et al., 2016). Additionally, by using the softmax function, our method to estimate IF can more easily be extended to multiclass problems.

3.2 Bayesian Neural Networks

In this section, we will formally introduce BNNs, a type of NN that is trained using Bayesian inference. We being by introducing the idea of uncertainty-aware deep learning before going on to formalise the theory central to understanding BNNs.

3.2.1 Uncertainty-Aware Deep Learning

Uncertainty quantification in deep learning represents an area of research that seeks to reduce the negative impacts of the various uncertainties underlying an algorithm. Methods developed in this area attempt to quantify the uncertainties associated with the accuracy, completeness, and selection of training data, among other sources (Abdar et al., 2021), and there are parallels between these sources of uncertainty and the sources of data bias described in Section 2.1.1. Bayesian techniques are one such method that have been explored to quantify uncertainty, however, the connection between Bayesian techniques in complex ML systems, such as NNs, and fairness has not previously been explored in the literature, thus we hope to address this gap in this work.

3.2.2 Applying Bayesian Inference Over Neural Networks

Probabilities are often interpreted using either frequentist or Bayesian statistics. With frequentist inference, the probability of an event occurring is based on the frequency of events. On the other hand, Bayesian inference determines the probability of an event occurring based on our previous knowledge of the events (Jospin et al., 2022). Bayes'

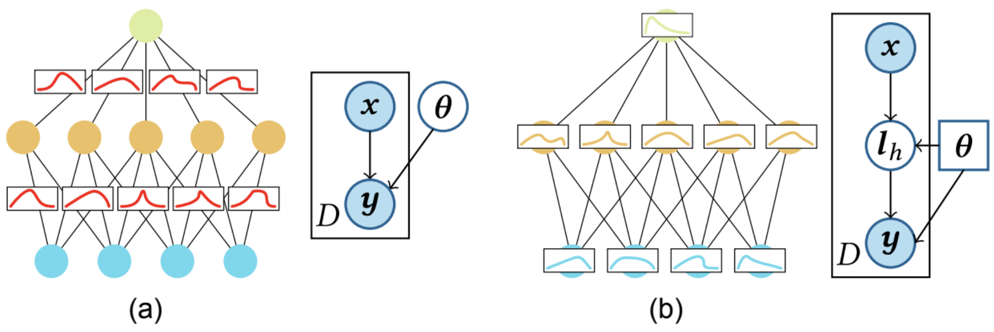

Figure 3.2: Diagram of a BNN with (a) stochastic weights and (b) stochastic activations. Image courtesy of Jospin et al. (2022).

theorem defines the posterior distribution of the parameters θ, given evidence X as:

$$p(\theta|X) = \frac{p(X|\theta)p(\theta)}{p(X)}, \tag{3.2}$$

where $p(\theta|X)$ is the posterior distribution, $p(\theta)$ is the prior distribution, $p(X|\theta)$ is the likelihood, and $p(X)$ is the marginal distribution.

A BNN is a type of stochastic NN that is trained by applying Bayesian inference over a NN (Jospin et al., 2022). In contrast to deterministic NNs that learn suitable weights and biases using processes such as backpropagation and gradient descent, BNNs assign the model stochastic activations or stochastic weights which simulate various models and the probabilities associated with each (Jospin et al., 2022). An example of a BNN is shown in Figure 3.2. Additionally, Figure 3.3 shows a side-by-side comparison of a deterministic NN and a BNN.

Formally, consider the NN defined in Section 3.1.1, $f^w : X \to Y$, with its weights and biases parametrised by w. For BNNs, we assume a prior distribution $p(w)$ that reflects our prior beliefs about the model's parameters (Wicker et al., 2021). Given a dataset $\mathcal{D}$, these prior beliefs are updated using Bayes' theorem (3.2) as more data is observed,

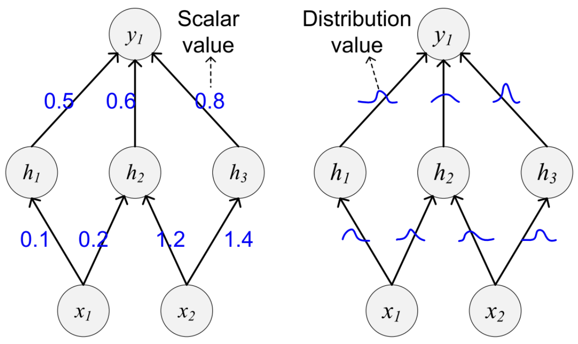

Figure 3.3: Side-by-side comparison of a deterministic NN with fixed scalar value weights (left) and a BNN with stochastic weights (right). Image courtesy of Jia et al. (2020).

to generate a posterior predictive distribution $p(w|\mathcal{D})$, such that:

$$p(w|\mathcal{D}) = \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})} \propto p(\mathcal{D}|w)p(w). \tag{3.3}$$

Note that the prior distribution $p(w)$ and likelihood $p(\mathcal{D}|w)$ are modelling choices.

The posterior predictive distribution, $p(w|\mathcal{D})$, represents a probability distribution over all possible model parameters, w, weighted by the likelihood of those parameters, given data, $\mathcal{D}$. Thus, by learning a probability distribution over all possible NN parameters, the posterior predictive distribution can be said to encode the uncertainty of a model's parameters (Cabiscol, 2019). If the various models tend towards agreeing in their predictions, we can say uncertainty is low and if they tend towards disagreeing in their predictions, we can say uncertainty is high (Jospin et al., 2022).

A prediction y for an unseen input x can be made in BNNs using the posterior predictive distribution as follows:

$$p(y|x,\mathcal{D}) = \int p(y|x,w)p(w|\mathcal{D})\mathrm{d}w. \tag{3.4}$$

This allows for the uncertainty of a model's parameters to be propagated through to the model's predictions (Cabiscol, 2019).

From this point forward, we use $\pi(x)$ to refer to the posterior predictive distribution over an input x, i.e the prediction made by a BNN for an input x, using the softmax function, σ, as the likelihood, such that:

$$\pi(x) = \int \sigma(f^w(x))p(w|\mathcal{D})\mathrm{d}w. \tag{3.5}$$

3.2.3 Inference (Training) Algorithms

In the context of BNNs, the process of applying Bayesian inference over a NN can be thought of as the training of a BNN. Determining an exact posterior predictive distribution is computationally infeasible for complex models such as NNs. Rather, in practice, approximations are used. Broadly, there are two main approaches to approximating the posterior predictive distribution: Variational Inference (VI) and Markov Chain Monte Carlo (MCMC) methods (Jospin et al., 2022). MCMC is often viewed as the gold standard for approximating the posterior, however, these methods are inefficient and not scalable, especially for complex models such as NNs (Khan et al., 2018). VI on the other hand offers a more scalable solution, therefore, in this project, we use VI to approximate the posterior predictive distribution.

Unlike MCMC methods, VI does not allow for sampling from the exact posterior predictive distribution, rather, it relies on a variational distribution with parameters that are learned to be as close as possible to the true posterior distribution (Jospin et al., 2022). Although VI is more scalable than MCMC, most methods are computationally and memory expensive. In order to address these limitations, we use a VI method known as Variational Online Gauss-Newton (VOGN), which offers a solution that is less demanding but comparable in quality (Khan et al., 2018). VOGN works by approximating the true predictive posterior distribution over the weights with a Gaussian distribution.

3.2.4 Advantages of Bayesian Neural Networks

The advantages of using BNNs over deterministic NNs include their ability to prevent overfitting, accurately learn from very small datasets, and quantify how certain or uncertain the model is about its predictions (Jospin et al., 2022). These advantages address many of the shortcomings of deterministic NNs which are prone to overfitting and usually require large amounts of training data to make accurate predictions. Additionally, a BNN's ability to model uncertainty is especially important when looking at fairness in ML, where incorrect predictions can have severe consequences. In comparison, when asked to make a prediction based on unexpected data points or patterns, a deterministic NN will tend to generalise and output an overconfident prediction as they do not have any mechanism to describe its uncertainty over the prediction.

3.3 Adversarial Examples

An adversarial example can be understood as an input intentionally designed by an attacker to cause a NN to misclassify the input. The phenomenon was first observed by Szegedy et al. (2013) in the context of image classification, in which they were able to cause a deterministic NN to misclassify an input image by applying a perturbation so small that it was imperceptible to the human eye.

3.3.1 Adversarial Attacks and the Fast Gradient Sign Method

There are many proposed methods to generate adversarial examples, and the reader is pointed towards Xu et al. (2020) for a review of these. These methods are called adversarial attacks. One such method is the Fast Gradient Sign Method (FGSM) first introduced by Goodfellow et al. (2014). Given an input x, FGSM works by first taking the gradient of the loss function L with respect to x. The sign of this gradient is then multiplied by an epsilon value to ensure the perturbation is small. The resulting perturbation is added to x to create an adversarial example x'. Formally, FGSM can be used to generate an adversarial example x' as follows:

$$x' = x + \epsilon \cdot \text{sign}(\nabla_x L(\theta, x, y)), \tag{3.6}$$

where θ is the model's parameter, x is the input to the model, y is the true label of the input, and $L(\theta, x, y)$ is the cost to train the NN (Goodfellow et al., 2014).

As its name suggests, FGSM is a fast method to generate adversarial examples as it only requires calculating one step of gradient descent. Thus, it is often used when large numbers of adversarial examples need to be generated, such as when adversarial examples need to be generated for all samples in a training set during adversarial training (Xu et al., 2020).

3.3.2 Adversarial Robustness

Adversarial robustness is a measure of a NN's ability to make correct predictions when presented with adversarial examples. In other words, a NN has high adversarial robustness if adversarial examples tend not to be misclassified by the network. There are two main notions of adversarial robustness: local adversarial robustness and global adversarial robustness. Local robustness verifies a model's robustness within a local neighbourhood of the input whereas global robustness verifies a model's robustness to changes to any input in the domain (Katz et al., 2017). Interestingly, it has been shown that BNNs show greater adversarial robustness in comparison to deterministic NNs. This will be discussed in more detail in Section 6.2, when the potential connection between the IF of a BNN and its adversarial robustness is explained.

3.4 Formulating Individual Fairness

In the following section, we will formulate the notion of IF introduced in Section 2.4.2. We begin by defining what a sensitive attribute is before providing the ϵ-δ definition of IF and the associated similarity metric d_{fair} that will be used in this project.

3.4.1 Sensitive Attributes

Sensitive attributes, also known as protected attributes, are features that can be used to identify an individual's protected class status either implicitly or explicitly (Barocas et al., 2019, Friedler et al., 2019). Common examples of sensitive attributes include gender, race, sexual orientation, and disability. A decision is generally classified as discriminatory if it is based on one or more of these sensitive attributes. Often, these attributes are protected by law, and it is illegal to discriminate against them (European Union, 2012, Parliament of the United Kingdom, 2010). Additionally, proxies for sensitive attributes must be taken into consideration to avoid the introduction of indirect discrimination.

3.4.2 ϵ-δ Individual Fairness Definition

In this section, we formalise the ϵ-δ IF definition described previously in Section 2.4.2. As stated, we can say that two individuals are similar if $d_{\text{fair}}(x, x') \leq \epsilon$, where d_{fair} is an appropriate similarity metric, x and x' are the input features for a pair of individuals, and ϵ is an appropriate threshold. If the predictions for all similar individuals in the test space are at least δ-close, then we can say the model is individually fair.

Definition 1 (ϵ-δ Individual Fairness (John et al., 2020))**.** *Given $\epsilon \geq 0$ and $\delta \geq 0$. A model $f^w : X \to Y$ is said to be individually fair iff*

$$\forall x, x' \in X s.t.\ d_{fair}(x, x') \leq \epsilon \Rightarrow |f^w(x) - f^w(x')| \leq \delta,$$

that is, if the predictions for ϵ-similar individuals are at least δ-close.

Importantly, based on context-specific knowledge, John et al. (2020) propose splitting the input features into sets $S_1, ..., S_t$ with corresponding thresholds $\epsilon_1, ..., \epsilon_t$. If a threshold is set to ∞ this implies that a pair of input features for two individuals, x_i and x_i' can differ in any way. This project will build upon this idea by splitting the input features into a set of sensitive and non-sensitive features, where sensitive features will be assigned a threshold of ∞, and non-sensitive features will be assigned a context-appropriate threshold. Intuitively, this means that irrespective of how significantly their sensitive features differ (e.g. irrespective of someone's race), two otherwise similar individuals will be classified similarly.

We define $d_{\text{fair}}(x, x')$ as a weighted version of an ℓ_p metric to measure the similarity between two individuals x and x', such that:

$$d_{\text{fair}}(x, x') = \sqrt[p]{\sum_{i=1}^{n} \theta_i |x_i - x_i'|^p}. \tag{3.7}$$

For sensitive attributes, the weights θ_i are set to zero. All other features are weighted depending on their correlation to the sensitive attributes. Intuitively, this means that the distance (i.e. dissimilarity) between two individuals should not be affected by an individual's sensitive attributes. The application of this metric to the method developed in this project will be discussed in Chapter 4.

3.5 Summary

In this chapter, we have formally introduced the theory required to follow the remainder of this report. First, we introduced NNs, before explaining how Bayesian inference can be applied over a NN in order to obtain a BNN. The theory behind adversarial examples, as well as FGSM, a method to generate them, and adversarial robustness were then explained. FGSM is of particular interest as it forms the basis of the method proposed within this project to estimate IF in BNNs. Finally, we formalised the ϵ-δ-IF definition that this project relies upon.

Chapter 4

Methodology

In this chapter, we describe the method developed in this project to measure IF in BNNs. We begin in Section 4.1 by first employing the ϵ-δ-IF definition, previously defined in Section 3.4.2, to BNNs and highlighting the parallels, and importantly the key distinctions, between this IF definition and definitions of adversarial robustness. In Section 4.1.1, we reframe measuring IF in BNNs as an optimisation problem consisting of two nested maximisation problems. To tackle the inner problem, in Section 4.2 we introduce *Fair-FGSM*, a modification of the FGSM adversarial attack, to generate adversarial examples that can be used to measure IF. The main distinction between *Fair-FGSM* and the standard FGSM, is that *Fair-FGSM* has been modified to work with an appropriate similarity metric d_{fair}. The outer problem, as described in Section 4.3, simply seeks to find the maximum difference δ^* in the predictive posterior distributions between all similar individuals x and x' in the test space. Finally, we propose that if $\delta^* \leq \delta$, where δ is a context-defined threshold for what constitutes similar treatment, then the model is considered to be individually fair.

4.1 Adversarial Example Approach for Estimating Individual Fairness

We begin by adopting the ϵ-δ-IF definition given in Definition 1 into the context of BNNs.

Definition 2 (ϵ-δ Individual Fairness for BNNs). *Given $\epsilon \geq 0$ and $\delta \geq 0$. A BNN f^w with a predictive posterior distribution π is said to be individually fair iff:*

$$\forall x, x' \in X s.t.\ d_{fair}(x, x') \leq \epsilon \Rightarrow |\pi(x) - \pi(x')| \leq \delta,$$

that is, if the predictive posterior distribution of ϵ-similar individuals are at least δ-close.

Practically, the only difference between Definition 1 and Definition 2 is that when considering BNNs we are interested in the differences in the predictive posterior distribution π, rather than the prediction of f^w.

Note that when measuring the adversarial robustness of a predictive posterior distribution π for a given input x, one checks that:

$$\forall x' s.t.\ |x - x'| \leq \epsilon \Rightarrow |\pi(x) - \pi(x')| \leq \delta. \tag{4.1}$$

In other words, if every feature of x is at least ϵ-close to every corresponding feature of the adversarial example x', then $\pi(x)$ and $\pi(x')$ should be at least δ-close.

There are clear parallels between the definition of adversarial robustness presented in Equation (4.1) and Definition 2, however, there are some key distinctions to be made. Firstly, Equation (4.1) only considers local robustness, whereas in estimating IF, we wish to consider global robustness, i.e. changes to any variable in the input space. Secondly, in practice, adversarial attacks often use ℓ_∞ distances with a single ϵ value, whereas we wish to use a task-specific d_{fair} which requires multiple ϵ to capture how correlated features are with a sensitive attribute. Thirdly, with adversarial robustness, we look at the average change in the model's prediction, whereas with IF we want to look at the worst-case change in the model's prediction. Thus, in order to translate techniques that rely on using adversarial examples to measure adversarial robustness into a measure of IF, we need to account for these distinctions.

4.1.1 Reformulating ϵ-δ-IF for BNNs as an Optimisation Problem

We proceed by reformulating Definition 2 as an optimisation problem consisting of two nested maximisation problems:

$$\delta^* = \max_{x \in X} \max_{\substack{x' \in X \\ d_{fair}(x,x') \leq \epsilon}} |\pi(x) - \pi(x')|. \tag{4.2}$$

The inner maximisation problem identifies a point x' in the d_{fair}-neighbourhood around an input x that maximises the difference between the predictive posterior distributions for $\pi(x)$ and $\pi(x')$. Intuitively, it seeks to find an individual x', that leads to a maximum difference in treatment when compared to the given individual x, despite being similar, as determined by d_{fair}.

The outer maximisation problem finds the maximum difference in predictive posterior distributions between all similar pairs of individuals (x, x'). Put simply, δ^* represents the worst-case difference in treatment when all individuals in a test space are considered. By checking whether $\delta^* \leq \delta$, where δ is a context-defined threshold for what constitutes similar treatment, we can determine whether the BNN is individually fair.

The inner maximisation problem can be solved through the use of adversarial attacks. Specifically, we introduce *Fair-FGSM*, a modified FGSM that considers the differences between adversarial robustness and fairness described in Section 4.1. The outer maximisation problem can be solved by simply calculating the maximum difference observed in the predictive posterior between all pairs of individuals (x, x') in the test space. In the subsequent sections, we present the formulation of the inner and outer maximisation problems, describing them in Sections 4.2 and 4.3, respectively.

4.2 *Fair-FGSM* (Inner Problem)

In this section, we consider the inner maximisation problem introduced in Equation (4.2). As described in Section 3.3.1, FGSM is one such technique to generate adversarial examples that can be used to evaluate the adversarial robustness of a NN. In order for FGSM to be suitable for measuring IF in BNNs, we make two major alterations to the traditional FGSM, as described in the subsequent Sections 4.2.1 and 4.2.2. We refer to the resulting FGSM as *Fair-FGSM*.

4.2.1 FGSM for Bayesian Neural Networks

Firstly, the definition of FGSM described in Section 3.3.1 considers deterministic NNs. Thus, it must be adapted to be suitable for BNNs.

Using the notation introduced in Chapter 3, the worst-case adversarial perturbation x', given an input x and a maximum perturbation magnitude $\epsilon > 0$, in a BNN is defined as:

$$x' = \underset{x' :\, ||x'-x|| \leq \epsilon}{\arg\max}\ \mathbb{E}_{p(w|\mathcal{D})}\left[L(x', w)\right], \tag{4.3}$$

where $\mathbb{E}_{p(w|\mathcal{D})}$ represents the expected value, or mean, of the loss function L applied over all possible weights w. In other words, x' is an input in the ϵ-neighbourhood around x that maximises L (Bortolussi et al., 2022). If the original input x and the adversarially perturbated input x' are predicted differently by a model, we call x' an adversarial example.

If we consider the definition of FGSM introduced in Section 3.3.1 for deterministic NNs, the only difference in the Bayesian context is that the adversarial attack is done against the posterior predictive distribution. Therefore, we can define FGSM for BNNs as follows:

$$x' = x + \epsilon \cdot \text{sign}(\mathbb{E}_{p(w|\mathcal{D})}\left[\nabla_x L(x,w)\right]). \tag{4.4}$$

4.2.2 Generalise FGSM to Consider d_{fair}

Secondly, we generalise FGSM to use a task-specific similarity metric, d_{fair}, rather than an ℓ_∞ distance as is traditionally used. Practically, this means adapting the method to support the use of multiple ϵ in order to denote both sensitive and non-sensitive features and their level of correlation. The method developed in this project relies on using the ℓ_p weighted metric (3.7), described in Section 3.4.2, as our similarity metric d_{fair}:

$$d_{\text{fair}}(x, x') = \sqrt[p]{\sum_{i=1}^{n} \theta_i |x_i - x'_i|^p}. \tag{4.5}$$

The reader is reminded that intuitively, d_{fair} is designed to encode the similarity between two individuals, such that an individual's sensitive features should not affect how similar they are. For example, if two individuals x and x' only differ in their gender, then $d_{\text{fair}}(x, x') = 0$. As mentioned previously, in reality, proxy attributes are often present, and these should also be reflected in the similarity metric.

To illustrate the connection between d_{fair} and our use of adversarial examples, assume we only have one sensitive feature, gender, and it is the first feature, i.e. $x_1 = x_{\text{gender}}$. To enforce that two individuals x and x' are similar, i.e. $d_{\text{fair}}(x, x') \leq \epsilon$, for every input x we generate a corresponding x', where each feature is ϵ-close such that ϵ correlates to the weights given to each feature in Equation (4.5). For example, if x_1 is the sensitive feature, then we would generate an adversarial example x' such that $|x_1 - x'_1| \leq \epsilon_{\text{sensitive}}$, where $\epsilon_{\text{sensitive}} = \infty$. In practice, as we commonly standardise input features between 0 and 1, we can use $\epsilon_{\text{sensitive}} = 1$. This captures the notion that the sensitive feature for x and x' can vary infinitely, thus enforcing that $|x_1 - x'_1|$ should have a weight θ of 0 in the similarity metric, in other words, two individuals' sensitive features should not affect the similarity between them. For all other features, we generate an adversarial example x' such that $|x_i - x'_i| \leq \epsilon_{\text{non-sensitive}}$, given $i = 2, ..., n$, where $\epsilon_{\text{non-sensitive}}$ corresponds to the weight given to that feature in Equation (4.5). Intuitively, a smaller $\epsilon_{\text{non-sensitive}}$ corresponds to a feature with less correlation to a sensitive feature, and vice versa.

4.3 Maximum Difference in Treatment (Outer Problem)

In this section, we consider the outer maximisation problem introduced in Equation (4.2), which seeks to identify the maximum difference in the predictive posterior distributions for all pairs of similar individuals (x, x').

We first train a BNN f^w using VI to obtain the posterior predictive distribution π. We then iterate through every input x in the test space and generate a corresponding adversarial example x' using the *Fair-FGSM* previously described in Section 4.2. x and x' are then used to estimate the worst-case change in the model's output when all pairs of similar individuals are considered:

$$\delta^* = \max_{x \in X} |\pi(x) - \pi(x')|. \tag{4.6}$$

If δ^* is less than or equal to δ, where δ is a context-dependent threshold for what constitutes similar treatment, then we say that the model is individually fair. For classification, $|\pi(x) - \pi(x')|$ represents the difference in class probabilities.

More details regarding the metrics collected by our experimental analysis are explained in Section 5.2, but broadly, models which are individually fair should have δ-small maximum differences in class probabilities and should not change the predicted class when all similar individuals x and x' in the test space are considered.

4.4 Overall Formulation

The overall formulation for the method proposed to measure IF in BNNs is summarised in Algorithm 1. We begin by considering a BNN model f^w trained using VI to obtain a posterior predictive distribution π. The *Fair-FGSM* attack, described in Section 4.2, is then used to generate a corresponding adversarial example x' for every input x over X. Intuitively, (x, x') can be conceptualised as a pair of similar individuals, as determined by d_{fair}, in which x' is generated to maximise the difference in prediction (treatment) when compared to x. In other words, what x' can be generated in order to obtain the worst-case change in a model's output, whilst remaining similar to x? The posterior predictive distribution is then applied over all inputs x and x', and the difference in outputs for each pair of similar individuals (x, x') is calculated. The maximum of these differences, δ^*, represents the worst-case treatment displayed by a model. Finally, if δ^* is less than or equal to δ, where δ is a context-defined threshold for what constitutes similar treatment, then the model is considered to be individually fair.

4.5 Summary

In this chapter, the method developed in this project to measure IF in BNNs has been described. We begin by reformulating the ϵ-δ IF definition for BNNs as an optimisation problem consisting of two nested maximisation problems. The inner maximisation problem identifies a point x' in the d_{fair}-neighbourhood around an input x that maximises the difference between a BNN's predictive posterior distributions $|\pi(x) - \pi(x')|$. We propose *Fair-FGSM*, a variant of the traditional FGSM attack modified to consider a d_{fair} similarity metric, as a solution to the inner problem. The outer problem that follows simply finds the maximum difference, δ^*, in the predictive posterior distributions between all

Algorithm 1 Measuring ϵ-δ-IF in BNNs

Input: Model: f^w, Sensitive Attribute Threshold: $\epsilon_{\text{sensitive}}$, Non-Sensitive Attribute Threshold: $\epsilon_{\text{non-sensitive}}$, Similar Treatment Threshold: δ

```
1: $\mathcal{D} \leftarrow PreprocessData()$    ▷ Load dataset, convert to one-hot encoding, normalise data, flatten to 1D vector
2: $X, Y, X_{\text{train}}, Y_{\text{train}} \leftarrow SplitData(\mathcal{D})$
3: $\pi \leftarrow VOGN(f^w, X_{\text{train}}, Y_{\text{train}})$
4: for all $x \in X$ do
5:     $x' \leftarrow FairFGSM(x, \epsilon_{\text{sensitive}}, \epsilon_{\text{non-sensitive}})$    ▷ Inner maximisation problem
6: end for
7: for all $(x, x')$ do
8:     $\delta^* \leftarrow \max(|\pi(x) - \pi(x')|)$    ▷ Outer maximisation problem
9: end for
10: if $\delta^* \leq \delta$ then
11:     Model $f^w$ with posterior predictive distribution $\pi$ is individually fair
12: end if
```

similar individuals x and x' in the test space. Finally, we propose that if $\delta^* \leq \delta$, where δ is a context-defined threshold for what constitutes similar treatment, then the model is considered to be individually fair.

In the following chapter, we present the implementation of the experimental analysis conducted to evaluate the method proposed in this chapter.

Chapter 5

Implementation

The following chapter describes the implementation of the method introduced previously in Chapter 4. We begin in Section 5.1 with an explanation of the experiments run to generate approximations of IF in a variety of BNNs with varying model architectures and similarity metric parameters. The metrics collected during these experiments are then discussed in Section 5.2. Following this, in Section 5.3 details regarding the tools and libraries used to implement the experiments are provided. Finally, we conclude this chapter with a discussion regarding the challenges faced during the implementation process in Section 5.4.

5.1 Experiments

In order to empirically evaluate the effectiveness of the method developed in Chapter 4, a series of experiments were run. Our experiments use our method to generate estimates of IF in BNNs with respect to various similarity metric parameters and model architectures. We also run the experiments on deterministic NNs, using the same parameters, to serve as a point of comparison.

5.1.1 Experimental Setup

The experiments are performed on the Adult dataset (Dua and Graff, 2017), a widely used benchmark task in the fairness literature (Ding et al., 2021, Fabris et al., 2022). This represents a binary classification problem that involves classifying whether an individual is predicted to make over \$50,000 a year, or not. The Adult dataset consists of 14 features which we divide into sensitive and non-sensitive attributes. We use gender as our sole sensitive attribute in our experiments. Other attributes include those that encode an individual's economic and educational status, such as occupation and highest degree of education. We use an 80 per cent training and 20 per cent test split. Additionally, all categorical features have been one-hot encoded, and all input features have been normalised to a value between 0 and 1.

Two ϵ similarity bounds are defined. We use $\epsilon_{\text{sensitive}} = \infty$ for the sensitive attribute across all experiments and test a variety of $\epsilon_{\text{non-sensitive}}$ values, ranging from 0.00 to 0.20 at intervals of 0.05, for non-sensitive attributes. As the features are standardised between 0 and 1, in our implementation we use $\epsilon_{\text{sensitive}} = 1$ rather than $\epsilon_{\text{sensitive}} = \infty$.

Additionally, a variety of model architectures are used. We experiment with varying the number of hidden layers from 1 to 5 and the number of neurons per layer across powers of 2 between 2 and 64. The choice of these ranges corresponds with those typically used for the Adult dataset in similar work (Yurochkin et al., 2020, John et al., 2020, Yeom and Fredrikson, 2020, Benussi et al., 2022). All other model parameters are standardised across experiments. The rectified linear unit (ReLU) activation function is used for all hidden layers, and the softmax activation function, described in detail in Section 3.1.2, is used for the output layer. Categorical cross-entropy is used as the loss function. The Adam optimiser (Kingma and Ba, 2014) is used for deterministic NNs and the VOGN optimiser (Khan et al., 2018) is used for BNNs. Finally, we set the batch size to 128 and the number of epochs to 15.

5.1.2 Details of Trials

For each of the model architectures and $\epsilon_{\text{non-sensitive}}$ values described in Section 5.1.1, a BNN and a deterministic NN are trained on the Adult dataset. Note that the BNN and deterministic NN are trained with the same hyperparameters as discussed in 5.1.1 to give $\pi(\cdot)$ and $f^w(\cdot)$, respectively. Additionally, the same training data is used for both the BNN and deterministic NN. The key difference here is that the BNN is trained using VI, specifically the VOGN optimiser, whereas the deterministic NN is trained using traditional methods of gradient descent.

For every input x in the test set, we generate a corresponding adversarial example x' using the *Fair-FGSM*, introduced in Section 4.2, that supports the use of multiple epsilons $\epsilon_{\text{sensitive}}$ and $\epsilon_{\text{non-sensitive}}$. The trained models are then used to generate predictions for all values of x and x'. For each model architecture and $\epsilon_{\text{non-sensitive}}$ value, we run ten trials and take the average of the results.

5.2 Metrics Collected

Our proposed method to estimate IF in BNNs relies on measuring the worst-case change in a model's prediction for two ϵ-similar individuals (4.5). However, in our experiments, we collect a variety of additional metrics for completeness. This section presents a discussion of these metrics.

5.2.1 Softmax Differences

We measure the exact difference in a model's predictions, or predictive posterior, for all x and x'. This is done by taking the absolute difference between the two softmax outputs for $\pi(x)$ and $\pi(x')$. In practice, as the softmax outputs represent probabilities for each class that add up to one and our experiments only deal with binary classification, we take the absolute difference between the two probabilities of an input belonging to the first class. For example given the model predictions $\sigma(x)_1 = 0.1$ and $\sigma(x)_2 = 0.9$ for $\pi(x)$, and $\sigma(x')_1 = 0.4$ and $\sigma(x')_2 = 0.6$ for $\pi(x')$, the difference in the model's prediction is $|0.1 - 0.4| = 0.3$. Note that this would be the same as taking the differences of the softmax outputs for the second class, i.e. $|0.9 - 0.6| = 0.3$. Intuitively, a difference of 0 means that the model's prediction was not influenced by the input's sensitive features, thus representing a fair model. The higher the value of the difference, the more a model's prediction was influenced by a sensitive feature, thus encoding greater levels of unfairness.

Given these differences, we compute a variety of metrics. As these metrics are computed as the difference between two softmax values, the values are bounded between 0 and 1, with 0 representing the fairest scenario, and 1 representing the least fair scenario.

Maximum Difference

$$\max_{x \in X} |\pi(x) - \pi(x')| \tag{5.1}$$

Our method proposes that a model π is individually fair if the worst-case difference in its predictions for all ϵ-similar individuals, x and x', is below a context-dependent threshold δ, as proposed in Definition 2. Therefore, comparatively, we can say models with a lower maximum difference in their predictions are more individually fair. This is the metric we rely on in the analysis of our results in Chapter 6, and the metric used to compare the IF of deterministic NNs and BNNs.

Mean Difference

$$\mu_{x \in X} |\pi(x) - \pi(x')| \tag{5.2}$$

As is often done when measuring adversarial robustness, we also calculate the mean across all differences in predictions for all ϵ-similar individuals, x and x'. However, we remind the reader that for individual fairness guarantees we wish to certify worst-case guarantees for any individual. This is because a model that mistreats an individual, even if the average treatment across all individuals is fair, is not individually fair. Due to this reasoning, we do not analyse this metric in our results. The interested reader instead is referred to Appendix A.2 for the experimental results relating to the mean differences.

Minimum Difference

$$\min_{x \in X} |\pi(x) - \pi(x')| \tag{5.3}$$

Additionally, we also calculate the minimum difference in predictions for all ϵ-similar individuals, x and x'. Again, for the same reasons as the mean difference, we do not analyse the minimum difference in predictions in our results. The reader is referred to Appendix A.3 for the experimental results relating to the minimum differences if interested.

5.2.2 Basic Individual Fairness Score

Given the outputs of our experiments, we also calculate a basic IF score based on the number of similar predictions made for all ϵ-similar individuals. As our experiments deal solely with classification, we simplify similar predictions here to predictions of the same class. This is calculated by counting the number of pairs of similar individuals (x, x') classified the same by a model and dividing it by the total number of pairs considered. For example, a basic IF score of 0.3 would mean that 30 per cent of the test inputs, x, were classified the same as their corresponding adversarial example, x'.

A higher basic IF score indicates a more individually fair model. However, it should be noted that as IF seeks to ensure worst-case guarantees across all individuals, the score outlined in Equation (5.1) is a more accurate measure of IF, and this basic score should serve only as a rough indicator of a model's fairness. Again, we do not analyse this metric in our results, however, the basic IF scores calculated during our experiments can be found in Appendix A.1.

5.3 Tools and Implementation Details

All experimental work was carried out on an Apple M1 chip with 8-core CPU, 7-core GPU, 16-core Neural Engine, 16 GB unified memory, and 512 GB SSD storage. All code was implemented and run using Python 3.9.12. The source code used to implement and run all experiments described in this project can be found at: `https://github.com/alicedoherty/bayesian-individual-fairness`.

5.3.1 Keras and TensorFlow

Keras (Chollet et al., 2015) is a deep learning API that acts as a wrapper to the machine learning platform TensorFlow 2 (Abadi et al., 2015). It is one of the most popular ML frameworks and is used both for personal projects and by well-known companies such as YouTube, Netflix, and Twitter (Chollet et al., 2015). Keras was used to implement the NNs used in our experiments. Listing 5.1 shows an example of how a simple deterministic NN can be implemented using Keras.

```
from tensorflow import keras
model = Sequential([
    Input(shape=(100,1)),
    Dense(64, activation="relu"),
    Dense(64, activation="relu"),
    Dense(2, activation="softmax"),
])
model.compile(loss=keras.losses.CategoricalCrossentropy(),
    optimizer=keras.optimizers.Adam(), metrics=["accuracy"])
model.fit(x_train, y_train, batch_size=128, epochs=15)
model.evaluate(x_test, y_test)
model.predict(x_test)
```

Listing 5.1: Example of a basic deterministic NN implemented using the Keras API.

5.3.2 Deepbayes

Keras does not provide built-in functionality for the training of BNNs. As such, in our experiments, we rely on Deepbayes (Wicker, 2022), a Python package that offers built-in support for the training of BNNs. Deepbayes is implemented using TensorFlow 2 and uses Keras as its interface. This allows for Keras models to be trained using Bayesian methods easily. Listing 5.2 shows an example of how a simple BNN can be implemented using Deepbayes.

```
from tensorflow import keras
import deepbayes.optimizers as optimizers
model = Sequential([
    Input(shape=input_shape),
    Dense(64, activation="relu"),
    Dense(64, activation="relu"),
    Dense(num_classes, activation="softmax"),
])
bnn_model = optimizers.VariationalOnlineGaussNewton().compile(
    model, loss_fn=keras.losses.CategoricalCrossentropy(), batch_size
    =128, epochs=15)
bnn_model.train(x_train, y_train, x_test, y_test)
model.predict(x_test)
```

Listing 5.2: Example of a basic BNN implemented using Deepbayes.

Edits to Deepbayes

Although Deepbayes is compatible with Keras and offers support for adversarial attacks, it is not as widely used and documented as other well-established frameworks, such as Keras. As a result, a number of minor edits were made to the source code in order to successfully run the experiments described in Section 5.1. The changes made to the Deepbayes source code are listed below. Additionally, the source code for the modified Deepbayes package can be found at: `https://github.com/alicedoherty/bayesian-individual-fairness/tree/main/deepbayes`.

- As the experiments were run on an M1 Mac, dependencies had to be updated in order to support `tensorflow-macos` rather than `tensorflow`.
- As the input data used in our experiments were one-hot encoded, the code had to be extended to support categorical accuracy and cross-entropy, rather than just sparse categorical accuracy and cross-entropy.
- Some of the code that relied on NumPy was outdated and had to be updated as a result.

5.3.3 *Fair-FGSM* Implementation

For the training of the models used in our experiments, we rely on the built-in optimisers and loss functions provided by Keras and Deepbayes. However, these are not suitable for the *Fair-FGSM* proposed in this project. Therefore, in order to support the modifications outlined in Section 4.2, we had to rely on developing custom code for FGSM, rather than using existing implementation solutions.

The code used to implement *Fair-FGSM* is outlined in Listing 5.3. This code was adapted from the FGSM implementation provided in the Deepbayes source code. The parameters to the function consist of the trained model π (`model`), the input data x (`inp`), the loss function L (`loss_fn`), and an array of ϵ values (`eps`). We highlight that this implementation utilises an *array* of ϵ, where each ϵ corresponds to a different feature of x. To illustrate, one input point x in the Adult dataset contains 100 features. The value of the feature at index 58 represents the gender of the individual (0 for female, 1 for male). Therefore, before calling `FGSM`, an array of length 100 is created where the value at index 58 is set to 1 ($\epsilon_{\text{sensitive}}$) and all other values are set to $\epsilon_{\text{non-sensitive}}$, which we vary between 0.00 and 0.20. This array is then passed to `FGSM` as the `eps` parameter, allowing us to apply a different ϵ to each feature.

The *Fair-FGSM* implementation, given in Listing 5.3 begins by setting the minimum and maximum vector bounds for each input feature, as determined by the feature's corresponding ϵ value (lines 5-8). The original prediction of the model for the given input x is then calculated (lines 11-14). Of particular interest are lines 16-19 which calculate the gradients of the loss function with respect to the input data. Specifically, we rely on TensorFlow's automatic differentiation functionality, provided by the `GradientTape` API to do this. The sign of the computed input gradient is then calculated (lines 22-23) and used to generate an adversarial example by multiplying it by the vector of ϵ values and adding it to the original input data (line 26). The adversarial example is then clipped to be a valid input and returned by the function.

```
1  def FGSM(model, inp, loss_fn, eps):
2      inp = inp.reshape(-1, inp.shape[0])
3  
4      # Set your max and min vector bounds
5      inp = np.asarray(inp)
6      vector_max = inp + eps
7      vector_min = inp - eps
8      inp = tf.convert_to_tensor(inp)
9  
10     # Get the original prediction you want to attack
11     temp = np.squeeze(model.predict(inp))
12     direction = np.zeros(len(temp))
13     direction[np.argmax(temp)] = 1
14     direction = direction.reshape(-1, direction.shape[0])
15 
```

```python
    with tf.GradientTape(persistent=True) as tape:
        tape.watch(inp)
        predictions = model.predict(inp)
        loss = loss_fn(direction, predictions)

    # Computed input gradient
    inp_gradient = tape.gradient(loss, inp)
    sign = np.sign(inp_gradient)

    # Add adversarial noise to the input
    adv = inp + eps*np.asarray(sign)

    # Clip it between your min and your max
    adv = np.clip(adv, vector_min, vector_max)

    # Clip it to be a valid input
    adv = np.clip(adv, 0, 1)

    return adv
```

Listing 5.3: Code for *Fair-FGSM* implementation.

5.3.4 Additional Libraries

Other additional Python libraries used for the implementation of this project include `pandas`, `numpy`, `matplotlib`, `seaborn`, and `scikit-learn`. These were used for the processing, analysis, and visualisation of data, as presented in Chapter 6.

5.4 Implementation Challenges

Due to the novelty inherent in this project's objectives and due to less work being done in the field of BNNs compared to deterministic NNs, there was an overall lack of resources available, making it more difficult to find solutions and debug errors. As a result, a number of implementation challenges were faced, as outlined below.

Training Bayesian Neural Networks As previously mentioned, for the training of BNNs the Deepbayes (Wicker, 2022) Python package was used. This choice was made due to its compatibility with the Keras interface and FGSM code. However, as this package is not consistently maintained and documented, a number of issues were encountered. Furthermore, as it is not a widely used package, no online resources were available to help solve the errors encountered. For example, some code was outdated and required edits in order to be usable. Additionally, as the experiments for this project were run on an Apple M1 chip, additional changes were required to ensure its compatibility. The edits made to the Deepbayes package are described in Section 5.3.2.

Generating Adversarial Examples with Multiple Epsilons As previously discussed, traditional implementations of FGSM only use one ϵ across all features. However, measuring IF using our developed method requires generating adversarial examples that support multiple ϵ values that can be used with different features. Consequently, existing tutorials and Python libraries for generating adversarial examples (e.g. Rauber et al. (2020)) could not be directly used. Ultimately, a custom implementation of FGSM, minorly adapted from the Deepbayes source code, was used to generate adversarial examples that support multiple ϵ values, as described in Section 5.3.3. Additionally, due to subtle differences in the implementation of deterministic NNs and BNNs two versions of the FGSM code were required.

Experimental Running Time Finally, due to the number of model architectures (30 total combinations) and similarity metric parameters (six in total) being tested, the training and computation time for the experimental analysis was significant. A set of deterministic NNs was also trained using the same parameters, doubling the number of experiments required to be run. Furthermore, ten trials of each experiment were run so that the average result could be taken. To illustrate, one experimental trial consisted of training a set of BNNs and deterministic NNs across all model architectures using a single $\epsilon_{\text{sensitive}}$ value and computing the resulting metrics described in Section 5.2. One of these trials would take approximately 2-3 hours to complete. These then had to be repeated for each of the six $\epsilon_{\text{sensitive}}$ values and ten trials, taking an estimated 120-180 hours to complete in total. In retrospect, a more efficient approach would have been to utilise a virtual machine, rather than running it locally and being restricted by personal time constraints. However, at the time, it was determined that the time and effort required to set up a virtual machine was not worth it.

5.5 Summary

In this chapter, specific details regarding the implementation of the method proposed in Chapter 4 and the experiments conducted to evaluate this method were presented. Specifically, we detail how our method was applied to measure IF in BNNs across a variety of model architectures and similarity metric parameters in order to evaluate its effectiveness. Details regarding the metrics collected during these experiments were also presented, the results of which are presented and analysed in the following chapter. Additionally, in this chapter, we provide details of the tools required in our implementation, such as Tensorflow, Keras, and Deepbayes, as well as the custom code developed to support our method. Finally, a discussion of the challenges faced whilst implementing the experiments was given.

Chapter 6

Results

In Chapter 5, we presented the experiments carried out to evaluate our proposed method for measuring IF in BNNs. In this chapter, we present the results of such experiments and discuss the conclusions that can be drawn from them. Importantly, as these results represent the first works in the literature measuring IF in BNNs it is difficult to compare and evaluate them against previous work. However, we provide a potential explanation for the results obtained by drawing on previous work done investigating adversarial robustness in BNNs. Finally, we provide a detailed discussion of the limitations of our work.

6.1 Analysis of Experimental Results

In order to validate the method developed to measure IF in BNNs, a series of experiments were conducted, as described in Section 5.1. This section will present and analyse the results of these experiments. Although a number of metrics were collected, as described in Section 5.2, the analysis in this section will focus on the maximum differences (5.1) obtained. This is because when measuring IF we want to measure the worst-case scenario across all individuals in an input space. However, the interested reader is referred to Appendix A for the mean differences (5.2), minimum differences (5.3), and basic IF scores recorded, although we highlight that these results cannot be used to draw useful conclusions about the IF of BNNs.

BNNs Are Empirically More Individually Fair Than Deterministic NNs We begin our analysis by visualising the maximum softmax differences between all pairs of ϵ-similar individuals, (x, x'), as a function of the number of hidden layers and width of a model. In Figure 6.1 it can be observed that for the vast majority of model architectures, BNNs are fairer than their deterministic counterparts. Additionally, in Figure 6.2 we plot the maximum softmax differences against the accuracy of the model. This shows the significant variance in IF estimates observed for deterministic NNs compared to BNNs across model architectures. From these results, we can conclude that although BNNs still display individual biases, they tend towards being fairer than deterministic NNs. Additionally, across varying model architectures it can be said that the range of IF estimates for BNNs is significantly smaller than that of deterministic NNs.

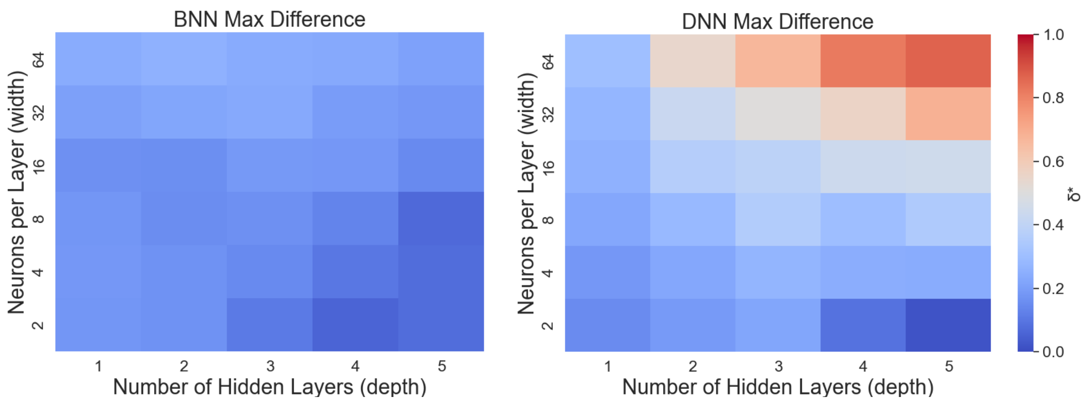

Figure 6.1: Maximum differences in softmax values for BNNs (left) and deterministic NNs (right), with respect to different model architectures (widths and depths), for $\epsilon_{\text{sensitive}} = 1.00$ and $\epsilon_{\text{non-sensitive}} = 0.00$.

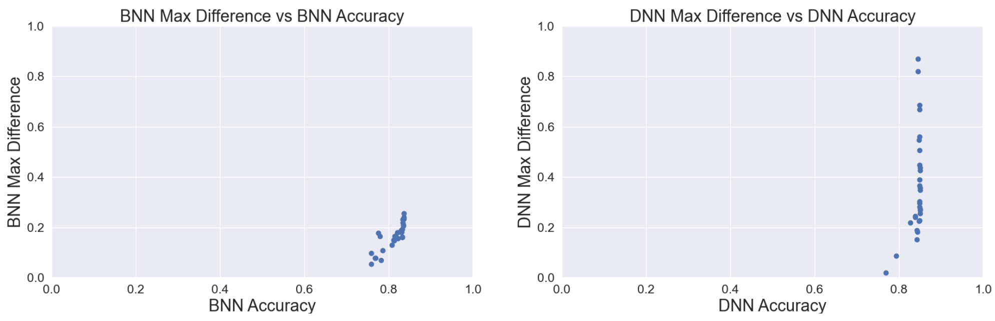

Figure 6.2: Maximum differences in softmax values plotted against the accuracy of the model for BNNs (left) and deterministic NNs (right) for $\epsilon_{\text{sensitive}} = 1.00$ and $\epsilon_{\text{non-sensitive}} = 0.00$.

BNNs Are More Robust to Increases in $\epsilon_{\text{non-sensitive}}$ When analysing the results of increasing $\epsilon_{\text{non-sensitive}}$, i.e. the similarity threshold for non-sensitive features, it can be observed that the decrease in fairness observed in BNNs is noticeably less than for deterministic NNs. The heat map in Figure 6.3 shows the maximum differences in predictions when $\epsilon_{\text{non-sensitive}} = 0.2$. Unsurprisingly, increasing $\epsilon_{\text{non-sensitive}}$ decreases the fairness across all model architectures as this allows for greater variation of non-sensitive attributes leading to greater changes in predictions. However, when compared to Figure 6.1, it can be seen that increasing $\epsilon_{\text{non-sensitive}}$ has a less significant impact on the fairness of BNNs compared to deterministic NNs. Figure 6.4 plots the change in maximum differences as $\epsilon_{\text{non-sensitive}}$ increases for both BNNs and deterministic NNs, for a model with 3 hidden layers and 8 neurons per layer. While both the fairness of BNNs and deterministic NNs decrease as $\epsilon_{\text{non-sensitive}}$ increases, the degree of change in BNNs is comparatively more stable and less pronounced.

Effect of Model Architecture on Individual Fairness Our experimental setup involved testing a number of different model architectures, more specifically, different numbers of hidden layers and neurons per layer. As shown in Figure 6.1 and Figure 6.3, as we increase the number of hidden layers and decrease the number of neurons per layer

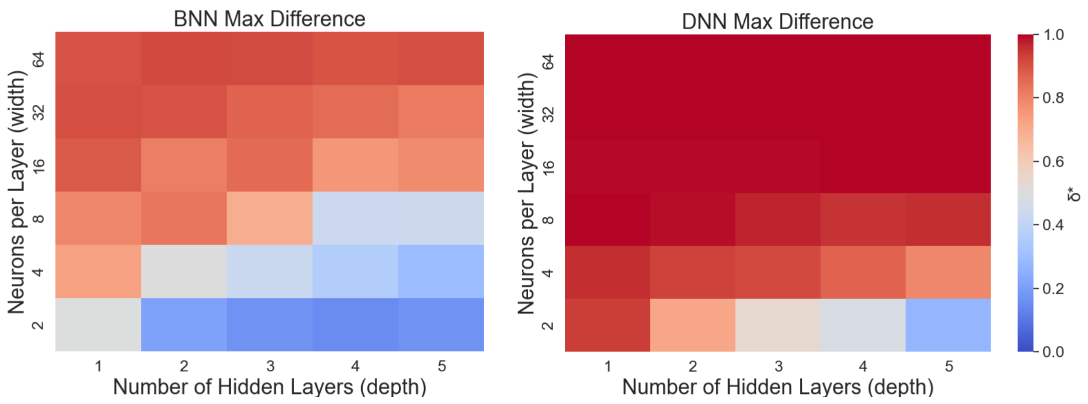

Figure 6.3: Maximum differences in softmax values for BNNs (left) and deterministic NNs (right), with respect to different model architectures (widths and depths), for $\epsilon_{\text{sensitive}} = 1.00$ and $\epsilon_{\text{non-sensitive}} = 0.20$.

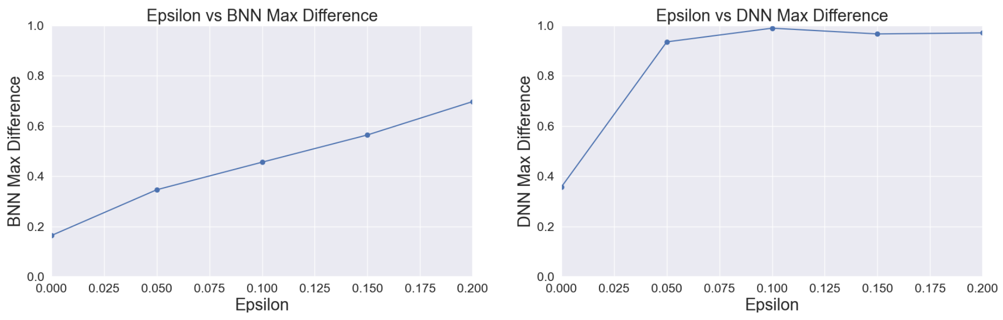

Figure 6.4: Maximum differences in softmax values plotted against $\epsilon_{\text{non-sensitive}}$ for BNNs (left) and deterministic NNs (right). The model used consists of 3 hidden layers and 8 neurons per layer.

in a model, fairness tends to increase, for both BNNs and deterministic NN. A similar trend was observed by Benussi et al. (2022) while evaluating their method for providing IF guarantees in deterministic NNs. Interestingly, the increased fairness as a model becomes deeper contrasts what has been observed with adversarial robustness, where increasing the number of hidden layers generally leads to more fragile models (Madry et al., 2017). This trend could perhaps be explained by the model becoming less reliant on specific input features, such as sensitive attributes, as it gets deeper, leading to a fairer model. Additionally, through the comparison of Figure 6.4 and Figure 6.5, it can be observed that the decrease in fairness for BNNs with more hidden layers and fewer neurons per layer (Figure 6.5), as $\epsilon_{\text{non-sensitive}}$ increases, is significantly less than those with less hidden layers and more neurons per layer (Figure 6.4).

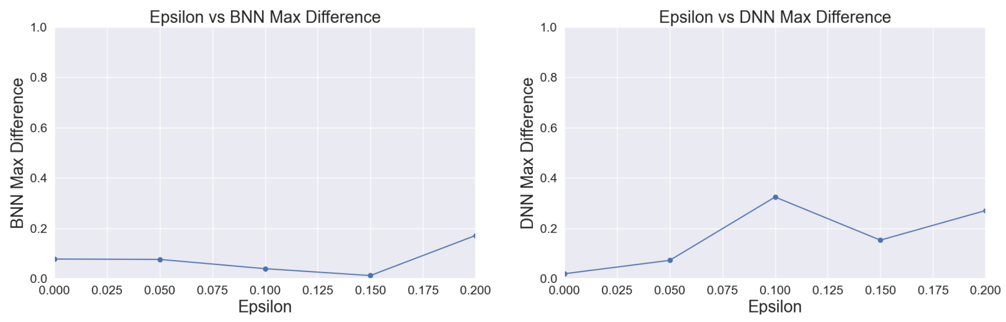

Figure 6.5: Maximum differences in softmax values plotted against $\epsilon_{\text{non-sensitive}}$ for BNNs (left) and deterministic NNs (right). The model used consists of 5 hidden layers and 2 neurons per layer.

Fairness-Accuracy Trade-Off As described in Section 2.2.5, the pursuit of fairness often comes at the cost of accuracy in deterministic NNs (Corbett-Davies et al., 2017). The results of our experiments show that this trend is also observed in BNNs. In Figure 6.6, it can be observed that BNN architectures with more hidden layers and fewer neurons per layer tend towards being fairer. This comes at the cost of accuracy as it can be seen in Figure 6.6 that BNN architectures with more hidden layers and a smaller width tend towards being less accurate.

6.2 Potential Explanation for Increased Fairness in Bayesian Neural Networks

The results of our experiments show that BNNs are empirically more individually fair than deterministic NNs. In this section, we provide a possible explanation as to why this may be the case.

As described in Section 3.2, the prediction made by a BNN is the combined average

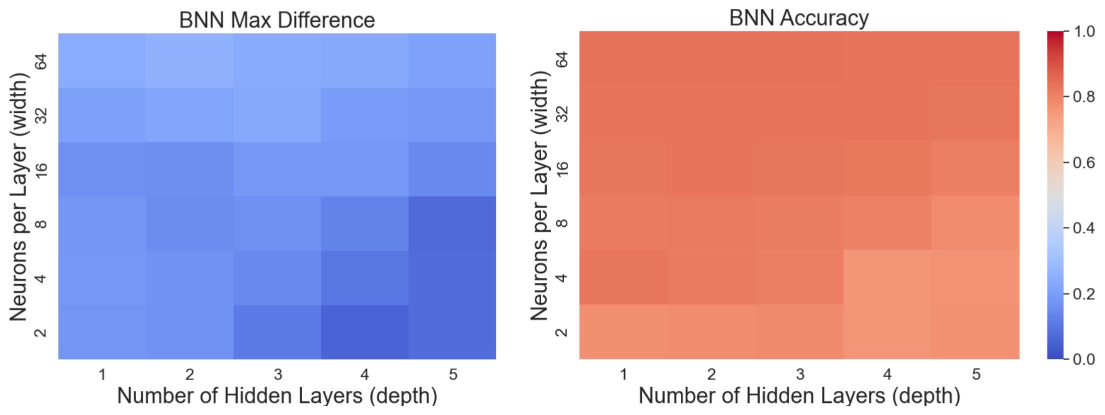

Figure 6.6: Maximum differences in softmax values (left) and accuracy (right) for BNNs, with respect to different model architectures (widths and depths), for $\epsilon_{\text{sensitive}} = 1.00$ and $\epsilon_{\text{non-sensitive}} = 0.00$.

of the results of multiple samples drawn from the posterior predictive distribution over the network's parameters. Specifically, in our experiments, 35 samples are used. It has been shown in the context of adversarial robustness that it is not a single pass through the trained network that is more robust to adversarial attacks but rather the average of multiple passes through the BNN that provides BNNs with greater adversarial robustness, compared to deterministic NNs (Bortolussi et al., 2022). Given the close relationship between our proposed method for evaluating IF and the notion of adversarial robustness, it is possible that the increased fairness of BNNs can be explained similarly. In other words, each sample NN drawn from the posterior predictive distribution of the BNN all display some form of bias but differ in such a way that they cancel out when averaged to give a prediction. If this is the case, we can say that BNNs learn fairer representations of data by averaging over multiple samples from the posterior predictive distribution.

6.3 Evaluation

This project set out to explore the feasibility of converting techniques for measuring adversarial robustness into a method for measuring IF in BNNs. The results of our experiments show that this is possible. Additionally, by applying our method to several BNNs and comparing the results to those of deterministic NNs, we have shown that BNNs are empirically more individually fair than deterministic NNs. These are novel outcomes that have not been presented previously in the literature.

As there are no previous results in the literature that measure IF in BNNs it is not possible to directly compare the results of our method to those of previous work. However, a number of steps were taken to ensure that the results obtained by our experiments were representative. Firstly, we used a range of model architectures, similar to those used in existing work done on IF and the Adult dataset (Yurochkin et al., 2020, John et al., 2020, Yeom and Fredrikson, 2020, Benussi et al., 2022). This provides greater insight into how the number of hidden layers or number of neurons per layer may be affecting a BNN's IF. Additionally, as our experiments were also run on deterministic NNs, the results for these can be loosely evaluated alongside previous work as a sanity check. Secondly, we also varied the $\epsilon_{\text{non-sensitive}}$ threshold used. Again, the values chosen align with those used in similar works (John et al., 2020, Benussi et al., 2022). This provides a much clearer picture as to the effects of this parameter on the IF of BNNs, allowing us to observe trends, such as the consequences of increasing $\epsilon_{\text{non-sensitive}}$ as discussed in Section 6.1. Furthermore, other standard experimental practices were followed, such as having a separate testing, training, and validation split. Additionally, for every variable tested, ten trials were run and the averaged results were used to ensure no significant outliers were present. Overall, results for thirty different model architectures, six different $\epsilon_{\text{non-sensitive}}$ thresholds, and ten trials for each combination of these were collected, providing a large amount of data to draw conclusions from.

6.4 Limitations

In this section, we will discuss several limitations to the method we have developed in this project and our experimental results. We remark that, as this is the first work evaluating IF in BNNs, in presenting this project we are not attesting to it being an all-encompassing and objective method, but rather it should be considered as a starting point for potential future work to build upon.

Not Accounting for Proxies The similarity metric used in our experimental analysis does not account for proxies. This is a significant limitation as the existence of proxies for sensitive attributes represents a far more realistic scenario than the one we have considered. Our experiments do account for a variation in the similarity threshold $\epsilon_{\text{non-sensitive}}$ for non-sensitive attributes, however, we only use a single value across all non-sensitive attributes. Rather it would be more realistic to have a different threshold for each non-sensitive attribute, similar to how (John et al., 2020) propose splitting input features into similar sets $S_1, ..., S_t$ each with a corresponding threshold $\epsilon_1, ..., \epsilon_t$, as described in Section 3.4.2. We note, however, that as the *Fair-FGSM* developed in this project supports the use of a different ϵ for every input feature, the similarity metric could be modified to account for proxies, using the intuition described in Equation (3.7) to encode how correlated a non-sensitive attribute is to a sensitive attribute. Additionally, as mentioned in Section 2.4.3, other methods to derive a similarity metric such as the Mahalanobis distance metric and feature embedding could be used. These offer a more accurate way to learn similarity metrics automatically from data and will be discussed further in Section 7.1 when commenting on potential future work.

Using a Single Sensitive Attribute Furthermore, only one sensitive attribute, gender, is accounted for in our experiments. However, the Adult dataset encodes other potentially sensitive attributes, such as race. Again, accounting for multiple sensitive attributes would represent a more realistic scenario. The method developed as it stands can currently account for multiple sensitive attributes by applying $\epsilon_{sensitive} = 1$ to all sensitive features in the input space, but due to time constraints, experiments were not run to evaluate this. Additionally, the sensitive attribute used was binary (i.e. gender could only be either male or female). However, in reality, sensitive attributes may not be binary, and may also not be mutually exclusive (e.g. an individual may be mixed-race). Again, we remark that our method can encode these situations but due to time constraints, experiments were not run to evaluate this.

Experimental Setup Our method was only evaluated on one benchmark fairness task, the Adult dataset. The motivation behind choosing this task was due to its popularity in the fairness literature(Barocas et al., 2019, Oneto and Chiappa, 2020, Pessach and Shmueli, 2022, Mehrabi et al., 2022). Despite its widespread use, it is not without its limitations. Concerns have been raised in relation to the age of the dataset and how representative the data is (Barocas et al., 2019, Ding et al., 2021). In using the Adult dataset, we are not attesting to the accuracy of the data or utilising it to make any real-world decisions. Rather, it is being used to explore the feasibility of the methods explored in this project to estimate IF. However, some have criticised the choice of the income threshold \$50,000 as being arbitrary and too high (Ding et al., 2021) which could affect the accuracy of the empirical findings of our experiments. However, as the results of our experiments were mainly considered in comparison to deterministic NNs, trained on the same task, rather than viewed as an objective measure of fairness we believe this is not a significant limitation. It would, however, be interesting to see how the results of our experiments would vary across different datasets and we propose this as a potential direction for future work in Section 7.1.

Additionally, we only evaluate a binary classification task. For multiclass classification problems, the method proposed could not be directly used, as currently only the difference between the probability of the first class is calculated. However, the method could be extended in future work to account for more than two class probabilities as proposed in Section 7.1.

6.5 Summary

In this chapter, we have presented the results of the experiments, described in Chapter 5, to measure IF in a variety of BNNs of varying model architectures and similarity metric parameters. Interestingly, we find that BNNs tend towards being more individually fair and show greater robustness to increases in $\epsilon_{\text{non-sensitive}}$ than their deterministic counterparts. Additionally, we observe that models with a greater number of hidden layers and fewer neurons per layer display greater individual fairness. We also find that the fairness-accuracy trade-off observed in deterministic NNs and other ML algorithms is also observed in BNNs. As no previous work has been done on measuring IF in BNNs, these results are non-trivial and represent the first of their kind in the literature. As such, we cannot provide a definitive reason as to why BNNs are fairer than deterministic NNs, but we posit that it is likely due reasons similar to those used to explain the greater adversarial robustness displayed by BNNs. We then discuss the considerations taken into account to ensure the accuracy and usefulness of our experimental analysis. Finally, we conclude this chapter by considering the limitations of our method and experimental results.

Chapter 7

Conclusions and Future Work

In this project, we set out to develop a method to evaluate IF in BNNs, the first work of its kind in the literature. The motivation to do so was driven, at a high level, by the increasing importance of developing fair deep learning algorithms during a time when ML is increasingly being deployed in situations that have a profound effect on both individuals and society as a whole. More specifically, we focus on BNNs due to their ability to model the uncertainty of their predictions and, as such, can offer a more trustworthy alternative to the more popular deterministic NN. Of particular interest is the increased robustness of BNNs to adversarial attacks (Bortolussi et al., 2022) given the connection between notions of adversarial robustness and IF, a relationship that has not previously been explored in the literature.

The core methodology developed and presented in this project relies on transferring techniques from measuring adversarial robustness into the context of the ϵ-δ-IF definition, which states that ϵ-similar individuals, as determined by a similarity metric d_{fair}, should be treated δ-similarly. We do this by framing the problem of measuring IF in BNNs as an optimisation problem. By extending FGSM, a widely used adversarial attack method, to use a similarity metric, d_{fair}, we propose *Fair-FGSM*. Using *Fair-FGSM*, we are able to generate a corresponding adversarial example x' for every individual x, such that x' represents an ϵ-similar individual that maximises the difference in the prediction of a BNNs for x and x'. In other words, we rely on adversarial examples to measure the worst-case treatment of two similar individuals, as defined by a context-specific similarity metric. By determining the worst-case treatment displayed by a model, when every individual is considered, we can determine whether a model is individually fair or not.

After establishing a method to measure IF in BNNs, we then use it to evaluate the IF of a range of BNNs that vary in architecture and similarity metric parameters. Although we recognise the limitations of our work, the results of our experiments interestingly show that, firstly, it is possible to utilise adversarial examples to measure IF in BNNs, and secondly, that BNNs tend to be more individually fair than deterministic NNs. These results represent the first of their kind in the literature and serve as an important starting point for future research in this area of growing interest and importance.

7.1 Future Work

As stated, this final year project has presented the first work on evaluating IF in BNNs, however, as discussed in Section 6.4 it is not without its limitations. In presenting our results, we remark that the methodology developed in this project is not proposed as an all-encompassing and objective solution, but rather should be viewed as a starting point for future work to build upon. Below, we recommend multiple areas of future work that could be explored to extend the work of this project.

Account for Proxies The similarity metric developed in this project did not account for proxies, however, the assumption that features can be treated as independent is not realistic. In Section 2.4.3, other methods to compute a similarity metric from data, that more accurately account for proxies, such as those based on the Mahalanobis distance

(Yurochkin et al., 2020) and feature embedding (Ruoss et al., 2020) were discussed. Additionally, work has been done to generate appropriate similarity metrics based on the judgement of human arbiters (Ilvento, 2019). These techniques have been applied to measure IF in deterministic NNs (Benussi et al., 2022), but have yet to be applied to BNNs, offering an interesting avenue for future work.

Multiple Sensitive Features and Intersectionality As noted when discussing this project's limitations, the experimental analysis in this project only considered a single sensitive feature. Clearly, it is far more realistic to consider multiple sensitive attributes at any one time when deploying deep learning systems in the real world. Although not experimentally tested, the method presented in this project can support multiple sensitive attributes, however, it would be interesting to empirically measure how the results vary as an increasing number of sensitive features are considered. Furthermore, and perhaps more interestingly, is that it has been observed that members of multiple sensitive groups experience a distinctive form of discrimination known as intersectionality (Crenshaw, 2017, Barocas et al., 2019). As this project has shown that BNNs are empirically more individually fair than deterministic NN it may be interesting to investigate whether this is also the case on tasks that have been shown to be susceptible to intersectionality, and if so by how much.

Extend to Multiclass Classification and Regression Tasks This project only analysed the results of measuring IF in BNNs on a single binary classification task. However, as our method is based on network gradients, it would be possible to extend the method to multiclass and regression tasks. As such, we propose the extension of the method developed in this project to support different types of tasks commonly used as fairness benchmarks. Additionally, given the criticisms against the Adult dataset described in Section 6.4, it would be interesting to see how BNNs and our proposed method perform across different tasks. Folktables (Ding et al., 2021), would especially be of interest as it is a dataset designed to address the shortcomings of the Adult dataset.

Connect Notions of Fairness and Explainable AI Traditional NNs are often treated as black-box models. This can often pose a problem when NNs are used to make high-stake or safety-critical decisions. The field of explainable AI, or interpretable AI seeks to develop methods to help humans better understand and interpret predictions made by NNs (Xu et al., 2019). Work has been done to exploit the increased transparency of BNNs to provide further explainability (Bykov et al., 2021). Although not directly related to measuring fairness, the increased explainability of BNNs presents an interesting parallel avenue of exploration. When considered alongside the potential for BNNs being fairer than deterministic NNs, as is suggested by the results of this project, this opens up the possibility to explore how BNNs can be exploited to make both fairer and more transparent decisions.

Fairness-Enhancing Mechanisms and Bayesian Neural Networks This project focused solely on measuring IF in BNNs. In Section 2.3.2, an alternate line of research within the fairness literature that seeks to develop fairness-enhancing mechanisms was introduced. No such work currently exists exploring how the IF of BNNs is affected by such mechanisms. Interestingly, as described in Section 2.3.2, adversarial training and GANs were primarily developed in the context of improving the adversarial robustness of NNs, but are now also being explored as a method to improve fairness in NNs (Zhang et al., 2018, Wadsworth et al., 2018, Xu et al., 2018, Celis and Keswani, 2019, Zhang and Sang, 2020). Given the close relationship between adversarial robustness and IF described in Section 4.1, it would be interesting to explore how the IF of BNNs is affected by these mechanisms.

7.2 Outlook and Reflection

In this project, we have explored IF in BNNs, the first work of its kind. Through the application of our developed method, we have shown that BNNs tend to be more individually fair than deterministic NNs. This represents an exciting first step towards the further exploration of the potential benefits associated with using BNNs as a fairer alternative to more established deep learning algorithms. Given the incredible importance of developing fair ML systems, we hope that in carrying out this project we have made a small contribution towards achieving this goal and that others may be inspired to pick up where we have left off.

On a personal note, the author would like to highlight that before undertaking this project, they had no prior formal knowledge of basic ML, let alone BNNs and adversarial techniques. As such, this project has presented a steep learning curve but nonetheless has provided a rewarding opportunity to explore a topic of great relevance and importance today.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X. (2015), 'TensorFlow: Large-scale machine learning on heterogeneous systems', `https://www.tensorflow.org/`. Software available from tensorflow.org.

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., Makarenkov, V. and Nahavandi, S. (2021), 'A review of uncertainty quantification in deep learning: Techniques, applications and challenges', *Information Fusion* **76**, 243–297.

Albarghouthi, A., D'Antoni, L., Drews, S. and Nori, A. V. (2017), 'Fairsquare: probabilistic verification of program fairness', *Proceedings of the ACM on Programming Languages* **1**(OOPSLA), 1–30.

Angwin, J., Larson, J., Mattu, S. and Kirchner, L. (2016), 'Machine bias', `https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing`.

Barocas, S., Hardt, M. and Narayanan, A. (2019), *Fairness and Machine Learning: Limitations and Opportunities*, fairmlbook.org. `http://www.fairmlbook.org`.

Bastani, O., Zhang, X. and Solar-Lezama, A. (2019), 'Probabilistic verification of fairness properties via concentration', *Proceedings of the ACM on Programming Languages* **3**(OOPSLA), 1–27.

Bellamy, R. K. E., Dey, K., Hind, M., Hoffman, S. C., Houde, S., Kannan, K., Lohia, P., Martino, J., Mehta, S., Mojsilovic, A., Nagar, S., Ramamurthy, K. N., Richards, J., Saha, D., Sattigeri, P., Singh, M., Varshney, K. R. and Zhang, Y. (2018), 'AI Fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias', *arXiv preprint arXiv:1810.01943* .

Benussi, E., Patane, A., Wicker, M., Laurenti, L. and Kwiatkowska, M. (2022), 'Individual fairness guarantees for neural networks', *arXiv preprint arXiv:2205.05763* .

Bogen, M. and Rieke, A. (2018), 'Help wanted: An examination of hiring algorithms, equity, and bias'.

Bortolussi, L., Carbone, G., Laurenti, L., Patane, A., Sanguinetti, G. and Wicker, M. (2022), 'On the robustness of Bayesian neural networks to adversarial attacks', *arXiv preprint arXiv:2207.06154* .

Buolamwini, J. and Gebru, T. (2018), Gender shades: Intersectional accuracy disparities in commercial gender classification, *in* S. A. Friedler and C. Wilson, eds, 'Proceedings of the 1st Conference on Fairness, Accountability and Transparency', Vol. 81 of *Proceedings of Machine Learning Research*, PMLR, pp. 77–91.

Byanjankar, A., Heikkilä, M. and Mezei, J. (2015), Predicting credit risk in peer-to-peer lending: A neural network approach, *in* '2015 IEEE Symposium Series on Computational Intelligence', IEEE, pp. 719–725.

Bykov, K., Höhne, M. M.-C., Creosteanu, A., Müller, K.-R., Klauschen, F., Nakajima, S. and Kloft, M. (2021), 'Explaining Bayesian neural networks', *arXiv preprint arXiv:2108.10346* .

Cabiscol, J. A. (2019), 'Understanding uncertainty in Bayesian neural networks', *Master of Philosophy (University of Cambridge)* .

Celis, L. E. and Keswani, V. (2019), 'Improved adversarial learning for fair classification', *arXiv preprint arXiv:1901.10443* .

Chollet, F. et al. (2015), 'Keras', `https://keras.io`.

Chouldechova, A. and Roth, A. (2018), 'The frontiers of fairness in machine learning', *arXiv preprint arXiv:1810.08810* .

Corbett-Davies, S., Pierson, E., Feller, A., Goel, S. and Huq, A. (2017), Algorithmic decision making and the cost of fairness, *in* 'Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', KDD '17, Association for Computing Machinery, New York, NY, USA, p. 797–806.

Crenshaw, K. (2017), *On Intersectionality: Essential Writings*, The New Press, New York, NY.

Dastin, J. (2018), 'Amazon scraps secret AI recruiting tool that showed bias against women', `https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G`.

De Fauw, J., Ledsam, J. R., Romera-Paredes, B., Nikolov, S., Tomasev, N., Blackwell, S., Askham, H., Glorot, X., O'Donoghue, B., Visentin, D. et al. (2018), 'Clinically applicable deep learning for diagnosis and referral in retinal disease', *Nature Medicine* **24**(9), 1342–1350.

Ding, F., Hardt, M., Miller, J. and Schmidt, L. (2021), 'Retiring Adult: New datasets for fair machine learning', *Advances in Neural Information Processing Systems* **34**, 6478–6490.

Dua, D. and Graff, C. (2017), 'UCI machine learning repository', `http://archive.ics.uci.edu/ml`.

Dwork, C., Hardt, M., Pitassi, T., Reingold, O. and Zemel, R. (2012), Fairness through awareness, *in* 'Proceedings of the 3rd Innovations in Theoretical Computer Science Conference', pp. 214–226.

European Union (2012), *Charter of Fundamental Rights of the European Union*, European Union, Brussels.

Fabris, A., Messina, S., Silvello, G. and Susto, G. A. (2022), 'Algorithmic fairness datasets: the story so far', *Data Mining and Knowledge Discovery* **36**(6), 2074–2152.

Feldman, M., Friedler, S. A., Moeller, J., Scheidegger, C. and Venkatasubramanian, S. (2015), Certifying and removing disparate impact, *in* 'Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 259–268.

Foulds, J. R., Islam, R., Keya, K. N. and Pan, S. (2020), Bayesian modeling of intersectional fairness: The variance of bias, *in* 'Proceedings of the 2020 SIAM International Conference on Data Mining', SIAM, pp. 424–432.

Friedler, S. A., Scheidegger, C. and Venkatasubramanian, S. (2016), 'On the (im) possibility of fairness', *arXiv preprint arXiv:1609.07236* .

Friedler, S. A., Scheidegger, C., Venkatasubramanian, S., Choudhary, S., Hamilton, E. P. and Roth, D. (2019), A comparative study of fairness-enhancing interventions in machine learning, *in* 'Proceedings of the Conference on Fairness, Accountability, and Transparency', FAT* '19, Association for Computing Machinery, New York, NY, USA, pp. 329–338.

Gajane, P. and Pechenizkiy, M. (2018), 'On formalizing fairness in prediction with machine learning', *arXiv preprint arXiv:1710.03184* .

Galhotra, S., Brun, Y. and Meliou, A. (2017), Fairness testing: testing software for discrimination, *in* 'Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering', pp. 498–510.

Goodfellow, I., Bengio, Y. and Courville, A. (2016), *Deep Learning*, MIT Press. `http://www.deeplearningbook.org`.

Goodfellow, I. J., Shlens, J. and Szegedy, C. (2014), 'Explaining and harnessing adversarial examples', *arXiv preprint arXiv:1412.6572* .

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2020), 'Generative adversarial networks', *Communications of the ACM* **63**(11), 139–144.

Hardt, M., Price, E. and Srebro, N. (2016), 'Equality of opportunity in supervised learning', *Advances in Neural Information Processing Systems* **29**.

Hébert-Johnson, U., Kim, M., Reingold, O. and Rothblum, G. (2018), Multicalibration: Calibration for the (computationally-identifiable) masses, *in* 'International Conference on Machine Learning', PMLR, pp. 1939–1948.

IBM (n.d.), 'What are neural networks?', `https://www.ibm.com/topics/neural-networks`.

Ilvento, C. (2019), 'Metric learning for individual fairness', *arXiv preprint arXiv:1906.00250* .

Jia, X., Yang, J., Liu, R., Wang, X., Cotofana, S. D. and Zhao, W. (2020), 'Efficient computation reduction in Bayesian neural networks through feature decomposition and memorization', *IEEE Transactions on Neural Networks and Learning Systems* **32**(4), 1703–1712.

John, P. G., Vijaykeerthy, D. and Saha, D. (2020), Verifying individual fairness in machine learning models, *in* 'Conference on Uncertainty in Artificial Intelligence', PMLR, pp. 749–758.

Jospin, L. V., Laga, H., Boussaid, F., Buntine, W. and Bennamoun, M. (2022), 'Hands-on Bayesian neural networks—a tutorial for deep learning users', *IEEE Computational Intelligence Magazine* **17**(2), 29–48.

Katz, G., Barrett, C., Dill, D. L., Julian, K. and Kochenderfer, M. J. (2017), Towards proving the adversarial robustness of deep neural networks, *in* L. Bulwahn, M. Kamali and S. Linker, eds, 'Proceedings of the First Workshop on Formal Verification of Autonomous Vehicles (FVAV '17)', Vol. 257 of *Electronic Proceedings in Theoretical Computer Science*, pp. 19–26. Turin, Italy.

Kearns, M., Neel, S., Roth, A. and Wu, Z. S. (2018), Preventing fairness gerrymandering: Auditing and learning for subgroup fairness, *in* 'International Conference on Machine Learning', PMLR, pp. 2564–2572.

Kearns, M. and Roth, A. (2019), *The Ethical Algorithm: The Science of Socially Aware Algorithm Design*, Oxford University Press, New York, NY.

Khan, M., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y. and Srivastava, A. (2018), Fast and scalable Bayesian deep learning by weight-perturbation in Adam, *in* 'International Conference on Machine Learning', PMLR, pp. 2611–2620.

Kingma, D. P. and Ba, J. (2014), 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .

Kleinberg, J., Mullainathan, S. and Raghavan, M. (2016), 'Inherent trade-offs in the fair determination of risk scores', *arXiv preprint arXiv:1609.05807* .

Lambrecht, A. and Tucker, C. (2019), 'Algorithmic bias? an empirical study of apparent gender-based discrimination in the display of stem career ads', *Management science* **65**(7), 2966–2981.

Liu, C., Arnon, T., Lazarus, C., Strong, C., Barrett, C., Kochenderfer, M. J. et al. (2021), 'Algorithms for verifying deep neural networks', *Foundations and Trends® in Optimization* **4**(3-4), 244–404.

Lohia, P. K., Ramamurthy, K. N., Bhide, M., Saha, D., Varshney, K. R. and Puri, R. (2019), Bias mitigation post-processing for individual and group fairness, *in* 'ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal processing (ICASSP)', IEEE, pp. 2847–2851.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D. and Vladu, A. (2017), 'Towards deep learning models resistant to adversarial attacks', *arXiv preprint arXiv:1706.06083* .

McNamara, D., Ong, C. S. and Williamson, R. C. (2017), 'Provably fair representations', *arXiv preprint arXiv:1710.04394* .

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K. and Galstyan, A. (2022), 'A survey on bias and fairness in machine learning', *ACM Comput. Surv.* **54**(6).

Mitchell, S., Potash, E., Barocas, S., D'Amour, A. and Lum, K. (2020), 'Prediction-based decisions and fairness: A catalogue of choices, assumptions, and definitions'.

Mukherjee, D., Yurochkin, M., Banerjee, M. and Sun, Y. (2020), Two simple ways to learn individual fairness metrics from data, *in* H. D. III and A. Singh, eds, 'Proceedings of the 37th International Conference on Machine Learning', Vol. 119 of *Proceedings of Machine Learning Research*, PMLR, pp. 7097–7107.

Oneto, L. and Chiappa, S. (2020), ‘Fairness in machine learning’.

Parliament of the United Kingdom (2010), ‘Equality act 2010’, `https://www.legislation.gov.uk/ukpga/2010/15`.

Pedreshi, D., Ruggieri, S. and Turini, F. (2008), Discrimination-aware data mining, *in* ‘Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining’, pp. 560–568.

Perrone, V., Donini, M., Zafar, M. B., Schmucker, R., Kenthapadi, K. and Archambeau, C. (2021), Fair Bayesian optimization, *in* ‘Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society’, pp. 854–863.

Pessach, D. and Shmueli, E. (2022), ‘A review on fairness in machine learning’, *ACM Comput. Surv.* **55**(3).

Rauber, J., Zimmermann, R., Bethge, M. and Brendel, W. (2020), ‘Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax’, *Journal of Open Source Software* **5**(53), 2607.

Ruoss, A., Balunovic, M., Fischer, M. and Vechev, M. (2020), Learning certified individually fair representations, *in* H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin, eds, ‘Advances in Neural Information Processing Systems’, Vol. 33, Curran Associates, Inc., pp. 7584–7596.

Saleiro, P., Kuester, B., Stevens, A., Anisfeld, A., Hinkson, L., London, J. and Ghani, R. (2018), ‘Aequitas: A bias and fairness audit toolkit’, *arXiv preprint arXiv:1811.05577* .

Schmidhuber, J. (2015), ‘Deep learning in neural networks: An overview’, *Neural Networks* **61**, 85–117.

Simonite, T. (2015), 'Probing the dark side of Google's ad-targeting system', https://www.technologyreview.com/2015/07/06/110198/probing-the-dark-side-of-googles-ad-targeting-system/.

Speicher, T., Heidari, H., Grgic-Hlaca, N., Gummadi, K. P., Singla, A., Weller, A. and Zafar, M. B. (2018), A unified approach to quantifying algorithmic unfairness: Measuring individual &group unfairness via inequality indices, *in* 'Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining', pp. 2239–2248.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. and Fergus, R. (2013), 'Intriguing properties of neural networks', *arXiv preprint arXiv:1312.6199* .

Tramer, F., Atlidakis, V., Geambasu, R., Hsu, D., Hubaux, J.-P., Humbert, M., Juels, A. and Lin, H. (2015), 'FairTest: Discovering unwarranted associations in data-driven applications', *arXiv preprint arXiv:1510.02377* .

Verma, S. and Rubin, J. (2018), Fairness definitions explained, *in* 'Proceedings of the International Workshop on Software Fairness', FairWare '18, Association for Computing Machinery, New York, NY, USA, pp. 1–7.

Wadsworth, C., Vera, F. and Piech, C. (2018), 'Achieving fairness through adversarial learning: An application to recidivism prediction', *arXiv preprint arXiv:1807.00199* .

Wicker, M. (2022), 'Deepbayes', https://github.com/matthewwicker/deepbayes.

Wicker, M., Laurenti, L., Patane, A., Chen, Z., Zhang, Z. and Kwiatkowska, M. (2021), Bayesian inference with certifiable adversarial robustness, *in* 'International Conference on Artificial Intelligence and Statistics', PMLR, pp. 2431–2439.

Xu, D., Yuan, S., Zhang, L. and Wu, X. (2018), FairGAN: Fairness-aware generative adversarial networks, *in* '2018 IEEE International Conference on Big Data (Big Data)', IEEE, pp. 570–575.

Xu, F., Uszkoreit, H., Du, Y., Fan, W., Zhao, D. and Zhu, J. (2019), Explainable AI: A brief survey on history, research areas, approaches and challenges, *in* 'Natural Language Processing and Chinese Computing: 8th CCF International Conference, NLPCC 2019, Dunhuang, China, October 9–14, 2019, Proceedings, Part II 8', Springer, pp. 563–574.

Xu, H., Ma, Y., Liu, H.-C., Deb, D., Liu, H., Tang, J.-L. and Jain, A. K. (2020), 'Adversarial attacks and defenses in images, graphs and text: A review', *International Journal of Automation and Computing* **17**(2), 151–178.

Yeom, S. and Fredrikson, M. (2020), 'Individual fairness revisited: Transferring techniques from adversarial robustness', *arXiv preprint arXiv:2002.07738* .

Yurochkin, M., Bower, A. and Sun, Y. (2020), Training individually fair ML models with sensitive subspace robustness, *in* 'International Conference on Learning Representations'.

Zehlike, M., Castillo, C., Bonchi, F., Baeza-Yates, R., Hajian, S. and Megahed", M. (2017), 'Fairness measures: A platform for data collection and benchmarking in discrimination-aware ml', `https://fairnessmeasures.github.io`.

Zeng, X., Dobriban, E. and Cheng, G. (2022), 'Bayes-optimal classifiers under group fairness', *arXiv preprint arXiv:2202.09724* .

Zhang, B. H., Lemoine, B. and Mitchell, M. (2018), Mitigating unwanted biases with adversarial learning, *in* 'Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society', pp. 335–340.

Zhang, Y. and Sang, J. (2020), Towards accuracy-fairness paradox: Adversarial example-based data augmentation for visual debiasing, *in* 'Proceedings of the 28th ACM International Conference on Multimedia', pp. 4346–4354.

Appendix A

Additional Experimental Results

In the following appendix, we present additional results collected as part of the experimental analysis described in Chapter 5. Specifically, we present the basic IF scores, the mean softmax differences, and minimum softmax differences collected for both the deterministic NNs and the BNNs tested in our experiments. Exact details regarding what these metrics represent and why they were not analysed in our results are provided in Section 5.2.

A.1 Basic Individual Fairness Scores

In Figure A.1, the basic IF scores calculated for both deterministic NNs and BNNs are visualised with respect to changes in the number of hidden layers and the number of neurons in a model, using $\epsilon_{\text{non-sensitive}} = 0.00$. Figure A.2 presents the same data but for the results of the experiments using $\epsilon_{\text{non-sensitive}} = 0.20$. We remind the reader that for the basic IF score a higher value is considered fairer, as described in Section 5.2. This is in contrast to the other metrics collected. Similar trends to those observed in Section 6.1 are seen in Figures A.1 and A.2. Specifically, through the analysis of the basic IF scores in BNNs and deterministic NNs, we observe that BNNs display a greater degree of fairness, BNNs show greater robustness to increases in $\epsilon_{\text{non-sensitive}}$, and models with a greater number of hidden layers and fewer neurons per layer are fairer than models with fewer hidden layers and more neurons per layer. As the basic IF score only considers changes in an input's overall predicted class, the metric is less sensitive in comparison to the maximum softmax difference, which considers exact class probabilities. This may explain why in Figure A.1 the differences in fairness between BNNs and deterministic NNs are less pronounced than in Figure 6.1. However, in Figure A.2, we observe that the differences in the basic fairness score, between BNNs and deterministic NNs, become more pronounced as $\epsilon_{\text{non-sensitive}}$ increases.

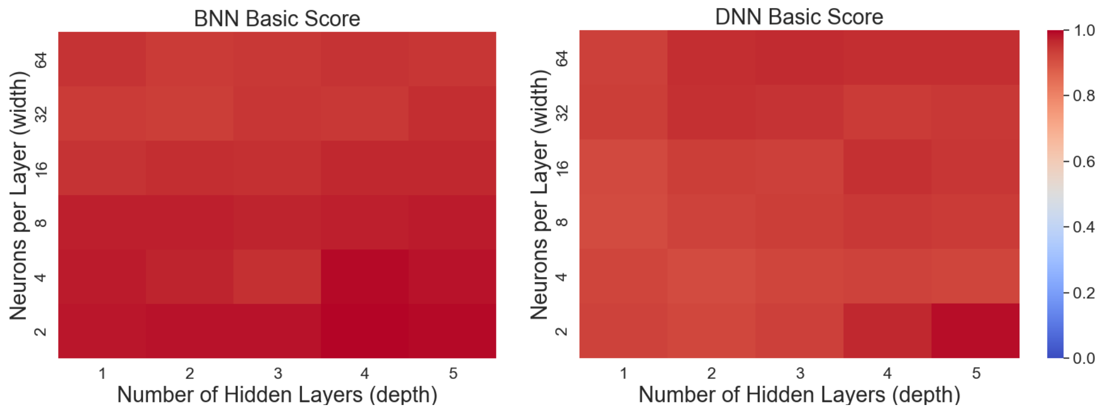

Figure A.1: Basic IF scores for BNNs (left) and deterministic NNs (right), with respect to different model architectures (widths and depths), for $\epsilon_{\text{sensitive}} = 1.00$ and $\epsilon_{\text{non-sensitive}} = 0.00$.

A.2 Mean Softmax Differences

In Figure A.3 the mean softmax differences calculated for both deterministic NNs and BNNs are visualised with respect to changes in the number of hidden layers and the

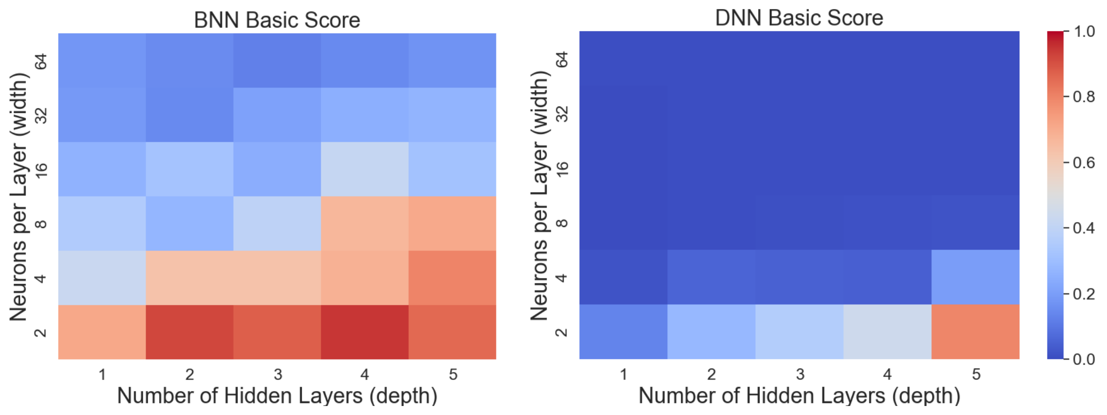

Figure A.2: Basic IF scores for BNNs (left) and deterministic NNs (right), with respect to different model architectures (widths and depths), for $\epsilon_{\text{sensitive}} = 1.00$ and $\epsilon_{\text{non-sensitive}} = 0.20$.

number of neurons of a model, using $\epsilon_{\text{non-sensitive}} = 0.00$. Figure A.4 presents the same data but for the results of the experiments using $\epsilon_{\text{non-sensitive}} = 0.20$. We observe that when considering $\epsilon_{\text{non-sensitive}} = 0.00$, the mean softmax differences are very similar in both deterministic NNs and BNNs. However, when $\epsilon_{\text{non-sensitive}} = 0.20$ is used, we observe that the differences in the mean softmax values for deterministic NNs and BNNs are more pronounced, namely that the mean softmax differences for deterministic NNs are much higher (less fair) than those for BNNs. Again, it is observed that BNNs show greater robustness to increases in $\epsilon_{\text{non-sensitive}}$, and models with a greater number of hidden layers and fewer neurons per layer are fairer than models with fewer hidden layers and more neurons per layer. Although, we remind the reader that, as explained in Section 5.2, taking the mean softmax differences does not give an accurate representation of the fairness of a model, as with IF we wish to give worst-case treatment guarantees, rather than average-case guarantees, with is closer in intuition to group fairness.

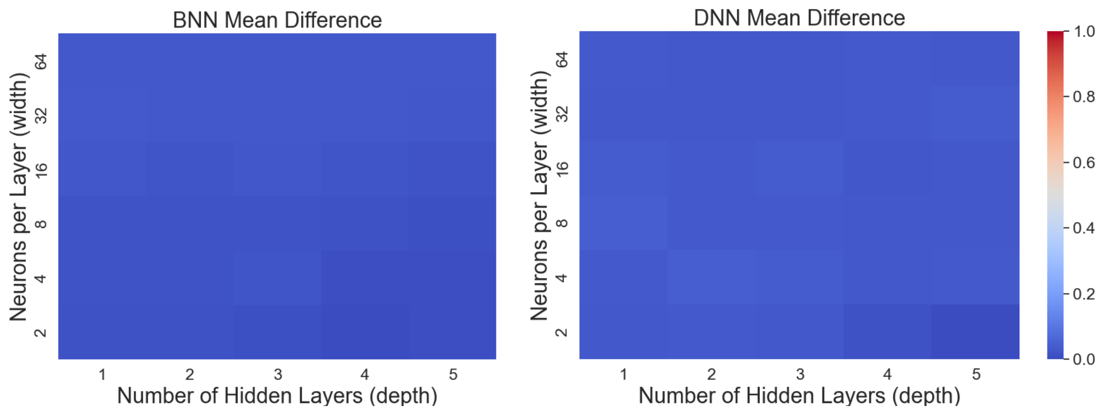

Figure A.3: Mean softmax differences for BNNs (left) and deterministic NNs (right), with respect to different model architectures (widths and depths), for $\epsilon_{\text{sensitive}} = 1.00$ and $\epsilon_{\text{non-sensitive}} = 0.00$.

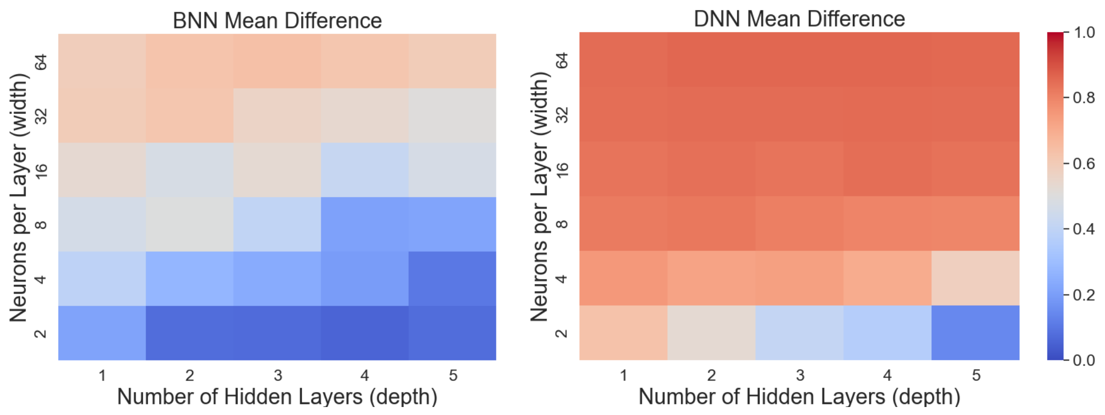

Figure A.4: Mean softmax differences for BNNs (left) and deterministic NNs (right), with respect to different model architectures (widths and depths), for $\epsilon_{\text{sensitive}} = 1.00$ and $\epsilon_{\text{non-sensitive}} = 0.20$.

A.3 Minimum Softmax Differences

Finally, in Figure A.5 the minimum softmax differences calculated for both deterministic NNs and BNNs are visualised with respect to changes in the number of hidden layers and the number of neurons of a model, using $\epsilon_{\text{non-sensitive}} = 0.00$. Figure A.6 presents the same data but for the results of the experiments using $\epsilon_{\text{non-sensitive}} = 0.20$. These results are presented for completeness, however, no conclusions of interest can be drawn from them. Intuitively, the minimum softmax differences could be thought of as a best-case treatment guarantee.

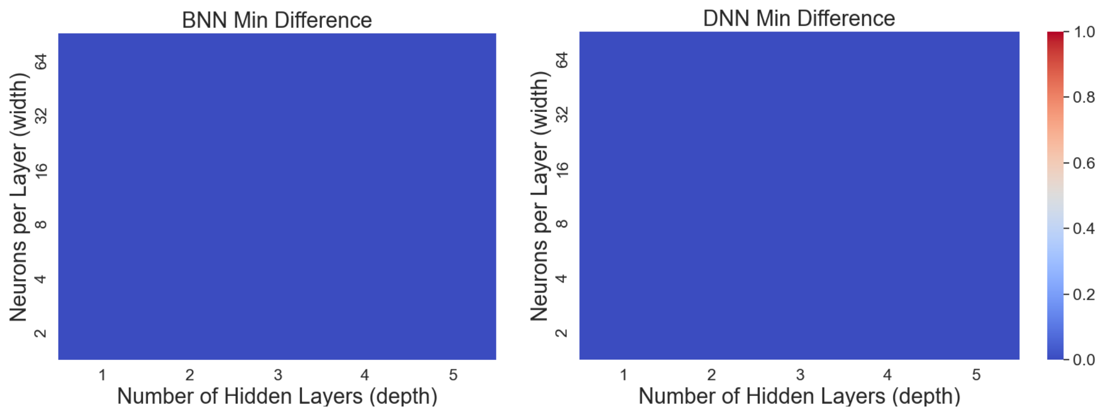

Figure A.5: Minimum softmax differences for BNNs (left) and deterministic NNs (right), with respect to different model architectures (widths and depths), for $\epsilon_{\text{sensitive}} = 1.00$ and $\epsilon_{\text{non-sensitive}} = 0.00$.

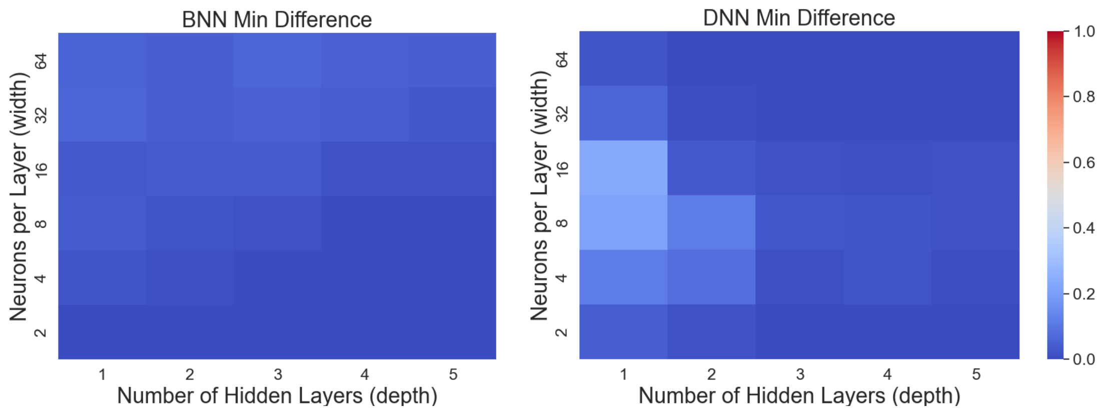

Figure A.6: Minimum softmax differences for BNNs (left) and deterministic NNs (right), with respect to different model architectures (widths and depths), for $\epsilon_{\text{sensitive}} = 1.00$ and $\epsilon_{\text{non-sensitive}} = 0.20$.